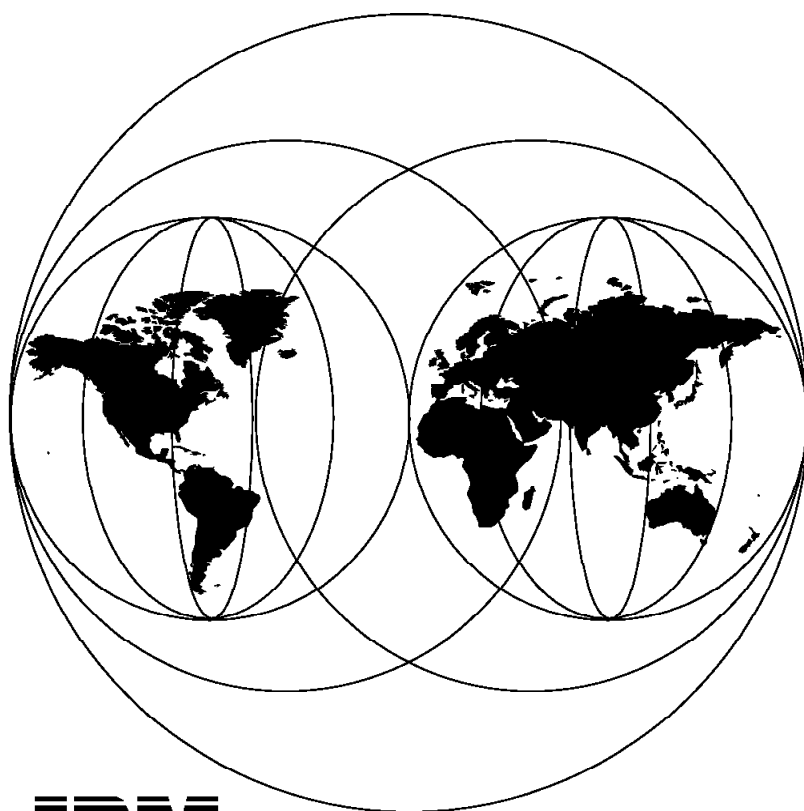


Customizing Performance Toolbox and Performance Toolbox Parallel Extensions for AIX

October 1997



**International Technical Support Organization
Austin Center**



International Technical Support Organization

SG24-2011-00

**Customizing Performance Toolbox and Performance
Toolbox Parallel Extensions for AIX**

October 1997

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix D, "Special Notices" on page 199.

First Edition (October 1997)

This edition applies to Version 2 of the Performance Toolbox for AIX and the Performance Aide for AIX Licensed Programs, as well as the Performance Parallel Extensions for AIX, a feature of IBM Parallel System Support Programs for AIX, Version 2. All these programs are for use with the AIX V4 Operating System.

Comments may be addressed to:

IBM Corporation, International Technical Support Organization
Dept. JN9B Building 045 Internal Zip 2834
11400 Burnet Road
Austin, Texas 78758-3493

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 1997. 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 | vii |
| Tables | xi |
| Preface | xiii |
| The Team That Wrote This Redbook | xiii |
| Comments Welcome | xv |
| Chapter 1. Introduction | 1 |
| 1.1 Suggested Study Methods | 2 |
| 1.2 How this Book is Organized | 2 |
| 1.3 Performance Toolbox Packaging | 3 |

Part 1. The Performance Toolbox 5

| | |
|--|----|
| Chapter 2. Manager - Basic Configuration | 7 |
| 2.1 The XMperf Resource File | 7 |
| 2.1.1 The XMperf Font | 7 |
| 2.1.2 The XMperf Colors | 9 |
| 2.1.3 The XMperf Shading | 10 |
| 2.1.4 The XMperf Patterns | 11 |
| 2.1.5 The XMperf Customizing | 11 |
| 2.1.6 XMperf LegendWidth | 13 |
| 2.2 The Rsi.hosts Host File | 13 |
| 2.3 The xmperf.cf Configuration File | 14 |
| 2.3.1 The xmperf Options | 15 |
| 2.3.2 The xmperf Console | 17 |
| 2.3.3 Console Keywords | 19 |
| 2.3.4 Local Instruments | 19 |
| 2.3.5 Remote Instruments | 23 |
| 2.3.6 Skeleton Instruments | 25 |
| 2.3.7 Menu Bar Commands | 33 |
| 2.4 The xmperf.hlp Help File | 45 |
| 2.4.1 Help Index | 50 |
| Chapter 3. Manager - Advanced Configuration | 51 |
| 3.1 The 3dmon Configuration | 51 |
| 3.1.1 The 3dmon Options | 53 |
| 3.1.2 The 3Dmon Resource File | 53 |
| 3.1.3 The 3dmon.cf Configuration File | 54 |
| 3.1.4 Customizing the 3dmon.cf File | 61 |
| 3.2 The 3dplay File | 65 |
| 3.2.1 The 3Dplay Configuration File | 65 |
| 3.2.2 The 3dplay.hlp Help File | 66 |
| 3.3 The Exmon Configuration | 66 |
| 3.3.1 The EXmon Resource File | 66 |
| 3.3.2 The exmon.cf Configuration File | 69 |
| 3.3.3 The exmon.hlp Help File | 70 |
| Chapter 4. Agent Configuration | 73 |
| 4.1 The xmserverd Configuration | 73 |

| | |
|--|-----------|
| 4.1.1 The xmservd.res File | 74 |
| 4.1.2 The xmservd.cf File | 76 |
| 4.1.3 The xmscheck Command | 79 |
| 4.2 The filtd Configuration | 81 |
| 4.2.1 The filter.cf Configuration File | 81 |
| 4.2.2 Creating Meaningful Alarms | 88 |
| 4.3 The SpmiResp Daemon | 89 |
| 4.3.1 The Resptime.cf File | 89 |
| 4.3.2 The iphosts Command | 89 |
| 4.4 The SpmiArmd Daemon | 90 |
| 4.4.1 The SpmiArmd.cf File | 90 |
| 4.4.2 The ARM Libraries | 90 |
| 4.4.3 Runtime Control | 91 |
| 4.5 The xmpeek Command | 91 |
| Chapter 5. Recording and Playback | 93 |
| 5.1 Recording | 93 |
| 5.1.1 Recording with xmperf | 93 |
| 5.1.2 Recording with 3dmon | 96 |
| 5.1.3 Recording with ptxrlog | 97 |
| 5.1.4 Recording with the xmservd Daemon | 99 |
| 5.2 Playback | 99 |
| 5.2.1 Playback with xmperf | 99 |
| 5.2.2 Playback with 3dplay | 103 |
| 5.2.3 Playback with azizo | 105 |
| 5.2.4 Using Annotations | 112 |

Part 2. The Performance Toolbox Parallel Extensions 113

| | |
|---|------------|
| Chapter 6. PTPE Overview | 115 |
| 6.1 How PTPE Works | 115 |
| 6.2 Using PTPE | 116 |
| 6.2.1 Data Collection and Data Archiving | 117 |
| 6.2.2 Data Access and Presentation | 117 |
| Chapter 7. Planning, Installing and Configuring PTPE | 119 |
| 7.1 Preparing Your SP for PTPE | 119 |
| 7.1.1 Which Nodes to Monitor | 119 |
| 7.1.2 Assigning Performance Roles to Nodes | 120 |
| 7.1.3 Choosing PTPE Statistics | 120 |
| 7.1.4 Choosing a PTX Manager | 121 |
| 7.1.5 Overview of Requisite Software | 121 |
| 7.2 Planning PTPE | 123 |
| 7.2.1 Planning a Hierarchy | 125 |
| 7.2.2 Preparing the Installation | 126 |
| 7.3 Installing PTPE | 126 |
| 7.3.1 Installing the Filesets | 127 |
| 7.3.2 Planning for PTPE Users | 128 |
| 7.3.3 Insuring Adequate Log and Archive Space | 129 |
| 7.3.4 AIX Syslog and Error Log | 129 |
| 7.3.5 SDR Entries for PTPE | 130 |
| 7.3.6 Pushing PTPE to Nodes through NIM | 130 |
| 7.4 Creating PTPE Hierarchy Specifications | 132 |
| 7.5 PTPE Command Interfaces | 134 |

| | |
|---|------------|
| 7.6 Preparing PTP Configuration Files | 134 |
| 7.6.1 Controlling PTP Data Collection | 135 |
| 7.6.2 Enabling PTX for PTP | 136 |
| 7.7 Summary | 137 |
| Chapter 8. Using PTP on SP Systems | 139 |
| 8.1 Verify PTP Operational Status | 140 |
| 8.2 Controlling PTP Hierarchy | 140 |
| 8.2.1 Modify PTP Hierarchy with Command Line Interface | 141 |
| 8.2.2 Modifying PTP Hierarchy with Perspectives | 141 |
| 8.2.3 Initializing PTP Hierarchy | 145 |
| 8.3 Controlling PTP Data Collection | 146 |
| 8.3.1 Starting PTP Data Collection | 146 |
| 8.3.2 Verifying Proper Operation | 147 |
| 8.3.3 Stopping PTP Data Collection | 147 |
| 8.4 Archiving Performance Data | 147 |
| 8.4.1 Starting Archiving and Collection Simultaneously | 148 |
| 8.4.2 Starting Archiving after Collection is Active | 148 |
| 8.4.3 Stopping Archiving without Stopping Collection | 148 |
| 8.5 Displaying Selected Statistics with Online PTX Consoles | 149 |
| 8.5.1 Using xmperf Consoles | 150 |
| 8.5.2 Using 3dmon Displays | 152 |
| 8.6 Offline Analysis of Archived Data | 153 |
| 8.6.1 Using ptpedump to Build ASCII files | 153 |
| 8.6.2 Using a2ptx to Create Recording Files | 154 |
| 8.7 What to Do When a Node Goes Down? | 155 |

Part 3. Monitoring and Analysis 157

| | |
|--|------------|
| Chapter 9. Monitoring and Analysis Hints | 159 |
| 9.1 Monitoring | 159 |
| 9.1.1 Monitoring with exmon | 159 |
| 9.1.2 Monitoring with 3dmon | 161 |
| 9.1.3 Monitoring with xmperf | 163 |
| 9.2 Analysis | 166 |
| 9.2.1 Postprocessing Recording Files | 166 |
| 9.2.2 Script to Postprocess Recording Files | 172 |
| 9.2.3 Spreadsheets | 172 |
| Appendix A. Sample Scripts | 173 |
| A.1 Sample Script to Create a Customized xmserverd.cf File | 173 |
| A.2 Sample Script to Split Recording Files | 178 |
| A.3 Regular Data Gathering with vmstat and Time Stamps | 181 |
| A.3.1 The vmit script | 181 |
| A.3.2 The format_vm Script | 181 |
| Appendix B. LMON: Developing a Local Monitor | 183 |
| B.1 Key Data Structures in Ichmon | 183 |
| B.2 Program Flow in Ichmon | 185 |
| B.2.1 Initialization | 185 |
| B.2.2 Collection | 187 |
| B.2.3 Termination | 188 |
| B.3 Making lmon | 188 |
| B.3.1 Input Format for a2ptx | 189 |

| | |
|---|------------|
| B.3.2 Initialization | 189 |
| B.3.3 Collection | 190 |
| Appendix C. Performance Toolbox (PTX) Fileset Contents | 193 |
| C.1 Manager | 193 |
| C.2 Agent | 194 |
| C.3 PTPE Filesets | 195 |
| C.3.1 The ptpc.program Image | 195 |
| C.3.2 The ptpc.gui Fileset | 195 |
| C.3.3 The ptpc.docs Fileset | 195 |
| C.4 PTPE Files | 195 |
| C.4.1 Files, Directories and Libraries | 196 |
| C.4.2 Commands and Utilities | 196 |
| C.4.3 Daemons | 197 |
| C.4.4 Message Catalogs | 197 |
| C.4.5 What PTPE Creates During Use | 198 |
| Appendix D. Special Notices | 199 |
| Appendix E. Related Publications | 201 |
| E.1 International Technical Support Organization Publications | 201 |
| E.2 Redbooks on CD-ROMs | 201 |
| E.3 Other Publications | 201 |
| How to Get ITSO Redbooks | 203 |
| How IBM Employees Can Get ITSO Redbooks | 203 |
| How Customers Can Get ITSO Redbooks | 204 |
| IBM Redbook Order Form | 205 |
| List of Abbreviations | 207 |
| Index | 209 |
| ITSO Redbook Evaluation | 211 |

Figures

| | | |
|-----|---|----|
| 1. | Main Menu - Default Font | 8 |
| 2. | Main Menu - Alias Font | 8 |
| 3. | Main Menu - Full String Font | 8 |
| 4. | The xmperf Color Selection Panel | 10 |
| 5. | XMperf Shading | 11 |
| 6. | XMperf Patterns | 11 |
| 7. | LegendWidth 8 | 16 |
| 8. | LegendWidth 32 | 16 |
| 9. | Default Mini Monitor | 18 |
| 10. | Local CPU Monitor Pull-down | 22 |
| 11. | Local CPU Monitor | 23 |
| 12. | Remote Memory Monitor Pull-down | 24 |
| 13. | Remote Memory Monitor | 25 |
| 14. | Local File Systems Pull-down | 26 |
| 15. | Selected Local File Systems | 26 |
| 16. | Local File System Console | 27 |
| 17. | Remote File System Pull-down | 28 |
| 18. | Host Selection - Remote File Systems | 29 |
| 19. | Remote File System Monitor | 29 |
| 20. | History Graphs Pull-down | 30 |
| 21. | History Graphs | 31 |
| 22. | Current Graphs Pull-down | 32 |
| 23. | Current Graphs | 33 |
| 24. | Who is Logged on Pull-down | 35 |
| 25. | The Who is Logged on Panel | 36 |
| 26. | The Who is Logged on Output Screen | 36 |
| 27. | Check FileSystem Sizes Pull-down | 37 |
| 28. | The Check FileSystem Sizes Panel | 37 |
| 29. | The Check FileSystem Size Output Screen | 38 |
| 30. | The iostat Options Pull-down | 40 |
| 31. | The iostat Options Panel - Complete | 40 |
| 32. | The iostat Options Output | 41 |
| 33. | File System Controls Pull-down | 42 |
| 34. | File System Controls - Increase File System Size Inputs | 42 |
| 35. | File System Controls - Increase File System Size Output | 43 |
| 36. | Controls Pull-down | 43 |
| 37. | Process Controls | 44 |
| 38. | Help Main Menu Pull-down | 45 |
| 39. | Help Panel | 46 |
| 40. | Help Console Pull-down | 46 |
| 41. | Help Main Window | 47 |
| 42. | Help On Console - History Graphs | 48 |
| 43. | FileSystem Size Help | 48 |
| 44. | Help On Command - iostat Options | 49 |
| 45. | Help On General - Delete Console | 49 |
| 46. | Help Index | 50 |
| 47. | 3dmon - Host Selection Made | 52 |
| 48. | 3dmon - Default Host Configuration | 52 |
| 49. | Disk Selection | 55 |
| 50. | 3dmon - Disk Configuration Output | 56 |
| 51. | 3dmon - Host disk Selection | 57 |

| | | |
|------|--|-----|
| 52. | 3dmon - disk Selection | 58 |
| 53. | 3dmon - disk | 59 |
| 54. | 3dmon - fs Selection | 60 |
| 55. | 3dmon - fs Hosts | 61 |
| 56. | 3dmon - CPU Selection | 62 |
| 57. | 3dmon - HOSTCPU CPU Output | 63 |
| 58. | 3dmon - multi host Selection | 64 |
| 59. | 3dmon - multi host Monitor | 65 |
| 60. | The Default exmon | 66 |
| 61. | The Modified exmon | 67 |
| 62. | Modified EXmon | 68 |
| 63. | Command Options of exmon | 70 |
| 64. | The exmon Main Help Panel | 71 |
| 65. | The filter.cf Command Output | 85 |
| 66. | The exmon Output from the cpubusy Alarm | 87 |
| 67. | Output from xmpeek | 92 |
| 68. | Partial Output from xmpeek -l | 92 |
| 69. | Example of Recording Pull-down Menu | 94 |
| 70. | File Exists Pop-up Window | 94 |
| 71. | Console Showing Tape Reels | 95 |
| 72. | Adding an Annotation Window | 95 |
| 73. | 3dmon Recording | 96 |
| 74. | Recording Selection Window | 97 |
| 75. | Output File Exists Window | 97 |
| 76. | Output from ptxrlog | 98 |
| 77. | Usage Syntax for mk.xmservd.cf | 99 |
| 78. | The xmperf Playback File Select | 100 |
| 79. | The xmperf Playback Window | 101 |
| 80. | The xmperf Playback Erase File | 102 |
| 81. | The xmperf Playback Seek | 102 |
| 82. | The xmperf Pop-up Playback Menu | 103 |
| 83. | 3dplay Select Recording File | 104 |
| 84. | 3dplay Playback | 105 |
| 85. | The azizo Recording File Pop-up Window from xmperf | 106 |
| 86. | The azizo Main Window | 107 |
| 87. | The azizo Main Window with Metrics | 108 |
| 88. | The azizo Graph Window | 109 |
| 89. | Zoom-in Dialog Box | 110 |
| 90. | Zoomed-in Graph of CPU Statistics | 110 |
| 91. | The azizo Filter Dialog Box | 111 |
| 92. | The azizo Info Window | 111 |
| 93. | Annotation List Window | 112 |
| 94. | Annotation View Window | 112 |
| 95. | Sample Two-Group PTPE Hierarchy for SP Monitoring | 116 |
| 96. | Multiple xmperf Instruments for Nodes 1, 3, and 5 | 118 |
| 97. | Script to Install perfagent.server and ptpе.program on a Node | 131 |
| 98. | Sample ptpеhier Input File to Define a Hierarchy with 16 Nodes | 133 |
| 99. | Hierarchy Displayed by ptpеhier -p | 133 |
| 100. | Using xmpeek -l to List PTPE Statistics Available | 135 |
| 101. | SP Environment for PTPE Testing | 139 |
| 102. | Note the Fully Qualified Hostnames in this Output From ptpеhier. | 140 |
| 103. | Perspectives Launch Pad with the Perfmon Tool in the Lower Right Corner | 142 |
| 104. | The Primary Perspectives Performance Monitor Interface | 143 |
| 105. | Node Options under Action Menu | 144 |

| | | |
|------|--|-----|
| 106. | Hierarchy Options under Action Menu | 144 |
| 107. | SDR Options under the Window Menu | 144 |
| 108. | New Options under the Utilities Menu after Integrating PTPE | 151 |
| 109. | New Consoles Added to the Monitor Menu | 152 |
| 110. | The exmon Window | 160 |
| 111. | The exmon Show File Window | 160 |
| 112. | The exmon Command Execution Window | 161 |
| 113. | 3dmon Host Selection | 162 |
| 114. | 3dmon lanresp Configuration Set | 162 |
| 115. | The xmperv Monitor Default Options | 163 |
| 116. | Output from ptxls | 167 |
| 117. | Output from ptxtab | 168 |
| 118. | Uncondensed Output from ptxhottab | 169 |
| 119. | Condensed Output from ptxhottab | 170 |
| 120. | Listing of Recording File before ptx2stat | 170 |
| 121. | The xmperv Playback before pt2stat | 171 |
| 122. | Listing of Recording File after ptx2stat | 171 |
| 123. | The xmperv Playback after ptx2stat | 172 |
| 124. | Sample Program to Simplify Ichmon.c Code | 183 |
| 125. | An Excerpt of the Variable legend[] | 184 |
| 126. | An Excerpt of the Definition of the Variable val1[] | 184 |
| 127. | Definition of prt1 with the Structure where | 185 |
| 128. | The Basic Sequence Used to Create and Define a StatSet | 186 |
| 129. | How Statistics Can be Added Dynamically | 186 |
| 130. | More Detail for Adding Statistics Dynamically | 187 |
| 131. | Sequence to Initiate and Maintain the Timed Collection of Data | 188 |
| 132. | Sample of a2ptx Input File | 189 |
| 133. | Code Initializing lmon Output: Hostname and Timestamp | 189 |
| 134. | Code to Print the Names of the Statistics we are Collecting | 190 |
| 135. | Simple Program to Verify an a2ptx Input | 190 |
| 136. | Code to Print the Timestamp. Placement is Critical | 191 |
| 137. | Code Added to Print the Actual Data Values | 192 |
| 138. | A Code Sample for Output Dependent on ValType of Statistic | 192 |

Tables

| | |
|--|-----|
| 1. AIX and PTX Levels | 121 |
| 2. Sample Table for Planning PTPE Hierarchy | 124 |
| 3. Sample Table for Planning Managers, Servers, and Coordinators | 124 |
| 4. Table of NIM Resources Defined to Ease Installation of PTP | 126 |
| 5. The PTPE GUI vs. the PTPE Command Line Interface | 134 |

Preface

This redbook is intended for anyone who needs to customize the monitoring and analysis of the Performance Toolbox for AIX (PTX). Especially for those who support one or more SP environments, there is a second part that covers an extension to PTX, appropriately named, the Performance Toolbox Parallel Extension (PTPE).

The customization and use of the Performance Toolbox (PTX) for AIX and Performance Toolbox Parallel Edition (PTPE) licensed products are thoroughly described.

We begin with a brief description of the basic components of the Performance Toolbox and PTPE. Part 1 covers customization of the standard PTX. Chapters 2, 3 and 4 describe in greater detail how the manager and agent components can be customized. Chapter 5 describes how you can use the Performance Toolbox to analyze performance statistics through run-time monitoring and/or through recording/playback.

Part 2 covers PTPE. This part also includes some introductory aspects not found in Part 1, such as planning and installation. Chapter 6 describes the conceptual framework of the Performance Toolbox Parallel Extensions (PTPE) in more detail. Chapter 7 reviews the basic issues and concepts regarding planning and installation of PTPE in an SP environment. Chapter 8 shows a number of examples of how PTPE actually behaves after having been installed.

Part 3 provides a short overview of monitoring and analysis concepts and tools.

Appendix A contains sample scripts; Appendix B shows how we changed the lchmon program to make it collect data into a file, and Appendix C provides a list of contents for the filesets mentioned in this book.

The Team That Wrote This Redbook

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

Andy Hoetzel is an International Technical Support Specialist for RS/6000 and AIX Performance at the International Technical Support Organization, Austin Center. He writes extensively and teaches IBM classes worldwide on all areas of AIX internals, performance and tuning. Andy holds a master of science degree in computer science from the University of Texas at El Paso. Before joining the ITSO, Andy Hoetzel worked in the AIX Competence Center in Munich, Germany as an AIX Technical Support Specialist.

Linda Bleikamp is an AIX Performance Consultant in the Performance Management and Capacity Planning National Competency Center. She is based out of Westlake, TX. Linda has been with IBM for eight years and has six years experience in system performance. Linda regularly presents performance topics and hands-on labs internally at IBM and at customer conferences throughout the United States. She holds a degree in computer science from Texas Christian University. Prior to joining IBM, she worked for eight years providing software technical support and managing a technical support team for another computer manufacturer.

Stephen Butcher is a Testing Specialist from Australia. He has two years experience in the performance testing area, for the last year concentrating on performance testing of Lotus Notes applications. He is currently in his second year of part-time study for a computer science degree.

Michael Felt is an AIX Certified Instructor and Lecturer in the Netherlands. He has a master's degree in Cognitive Psychology on software ergonomics and a minor in Computer Science from the Free University in Amsterdam. He has been active with UNIX since 1979 as system programmer specializing in performance, networking and porting. He is currently involved with the development of IBM world-wide course material for AIX.

Eric Jones is a Senior IT Specialist with the IBM Global Services Consulting and System Integration organization. He provides advanced services and consulting on SP infrastructure, system migration support, and parallel application enablement to companies with large SP systems. His recent focus has been on DB2 UDB implementations with SMP nodes for large data warehouse environments. Prior to joining IBM in 1995, he held a number of government and industry positions in the data management area.

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

Marcus Brewer
International Technical Support Organization, Austin Center

Tara Campbell
International Technical Support Organization, Austin Center

Steve Gardner
International Technical Support Organization, Austin Center

Jasenn McNair
International Technical Support Organization, Austin Center

Ryan France
IBM Austin

Stephen Nasypany
IBM Austin

Ann Ruth
IBM Austin

Barbara Wang
IBM Austin

Bob Gensler
IBM Poughkeepsie

Jim Chen
IBM Austin

Niels Christiansen
IBM Austin

Heidemarie Hoetzel
IBM Germany

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 211 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@vnet.ibm.com

Chapter 1. Introduction

This book is intended for anyone who is, or who must become, familiar with performance management. If that describes you, read on.

The content and organization of this book is based on the assumption that anyone interested in using this book is familiar with performance management issues. Equally important, we assume that you are already familiar with the graphical user interface of the Performance Toolbox - xmperf. As such, we don't discuss generic UNIX performance monitoring tools such as iostat, sar, or vmstat. Neither do we discuss performance tools specific to AIX included in the perfagent.tools fileset of the Performance Toolbox. Further, we assume that the perfmgr and perfagent filesets are already installed.

To verify that you do have these filesets installed, issue the following command on your AIX system:

```
lslpp -l perf*
```

If you don't get an indication that you have perfagent.server, perfagent.tools, perfmgr.common, and perfmgr.network (or (perfmgr.local) installed, make sure to install the perfmgr and perfagent filesets on your system.

So who are we writing for? Or, in other words, who do we think you are? We think that you are someone who needs to customize the monitoring and analysis of the Performance Toolbox for AIX (PTX). We therefore discuss the standard monitors, (automatic) data collection, doing more post-processing of data, and so on in the first part of the book. This should assist you regardless of whether you are reading this material out of curiosity or to fulfill a customer requirement.

Especially for those of you who support one or more Scalable POWERparallel (SP) environments, there is a second part that covers an extension to PTX, appropriately named, the Performance Toolbox Parallel Extension (PTPE). PTPE is an additional LPP (filesets ptpe.program and ptpe.gui) that can be purchased as an addition to the Parallel System Support Programs (PSSP) for system management of an SP environment.

Understanding the contents of Part 1 is very helpful, if not a requirement, for benefiting from Part 2. However, we don't assume that your knowledge of managing an SP is as great as your performance skills. There is the distinct possibility that you won't have to install PTPE yourself. However, we expect that the SP is installed and the nodes are operational. At a minimum, you need to give instructions about which nodes you want to have installed. Just in case you have to do it yourself, we supply information to make installing PTPE on an up-and-running Control Workstation (CWS) and its nodes very easy. Remember, PTPE is an additional product, and it might not have been installed on the CWS initially.

Review: After the introduction, the book is organized into two parts. Part 1 covers the customization of the standard Performance Toolbox Manager and Agent. Part 2 introduces Performance Toolbox Parallel Extension (hereafter PTPE). Assuming that your knowledge of performance is greater than your knowledge of managing an SP, we do provide some extra information for installing PTPE. This is the only part that should be quick reading for SP

specialists. But if your area is performance, you might find these hints helpful for the first couple of times you take a look at how well an SP is performing.

1.1 Suggested Study Methods

If you haven't used xmperf, then familiarize yourself with the graphical user interface. What we help you with in the first part of this book is finding out how to modify the graphical interface so that you can develop business solutions that meet customer requirements.

Once you are familiar with the manager, try to get some good recordings of performance statistics of a real load. The steps to learn are:

1. Record statistics and don't worry if there are too many being collected.
2. Analyze these statistics to determine which statistics are vital for monitoring the workload, possibly defining new ones from combinations of statistics.
3. Make new configurations so that only the vital statistics are collected.
4. Choose some thresholds and set up threshold trapping to user applications and/or network managers using the Simple Network Management Protocol (SNMP).
5. Define GUI displays for online or offline analysis of performance data.

1.2 How this Book is Organized

This book describes the customization and use of the Performance Toolbox (PTX) for AIX and Performance Toolbox Parallel Edition (PTPE) licensed products.

The rest of this chapter describes briefly the basic components of the Performance Toolbox and PTPE. Part 1 covers customization of the standard PTX. Chapters 2, 3 and 4 describe in greater detail how the manager and agent components can be customized. Chapter 5 describes how you can use the Performance Toolbox to analyze performance statistics through run-time monitoring and/or through recording/playback.

Part 2 covers PTPE. This part also includes some introductory aspects not found in Part 1, such as planning and installation. Chapter 6 describes the conceptual framework of the Performance Toolbox Parallel Extensions (PTPE) in more detail. Chapter 7 reviews the basic issues and concepts regarding planning and installation of PTPE in an SP environment. Chapter 8 shows a number of examples of how PTPE actually behaves after having been installed.

Part 3 gives a short overview of monitoring and analysis concepts and tools.

Appendix A contains sample scripts, and Appendix B shows how we changed the program lchmon into lmon, an application to provide ASCII statistics simultaneously to the screen display. Appendix C provides a list of contents for the filesets mentioned in this book.

This redbook is expected to be used in conjunction with other IBM publications or books similar to them. The following summary is a rough guide to books you should have available. Having a couple, if not all, of the performance-related books is required. The others (for instance, SP administration and planning-related books) are recommended.

The following books are standard IBM publications (white books) related to performance and tuning of an AIX system.

Performance Tuning Guide, SC23-2365

Performance Toolbox for AIX: Guide and Reference, SC23-2625

Performance Toolbox Parallel Extensions for AIX: Guide and Reference, SC23-3997

Other performance-related material from IBM includes:

RS/6000 Performance Tools in Focus, SG24-4989

Understanding IBM RS/6000 Performance and Sizing, SG24-4810

IBM Education & Training student notes for *AIX Performance Management* (Worldwide course code is AU28. Your country may catalog it under a different number, though).

Other IBM publications recommended are:

IBM PSSP 2.2 Administration Guide, GC23-3897

IBM PSSP 2.2 Installation and Migration, GC23-3898

IBM PSSP 2.2 Command and Technical Reference, GC23-3900

1.3 Performance Toolbox Packaging

The Performance Toolbox (PTX) for AIX consists of several components, each in a separate fileset. See Appendix C, "Performance Toolbox (PTX) Fileset Contents" on page 193, for a detailed listing of the filesets.

- Manager** Available in Local (1 host) or Network (n hosts) versions. That is, *perfmgr.common* plus either *perfmgr.local* or *perfmgr.network*. Unless otherwise specifically stated, we assume that you are using the network version.
- Agent** Known as Performance Aide (PAIDE/6000) or PTX client (*perfagent.server*). On an SP, this part of PTX is included in the *bos.obj.ssp* install objects. For example, *bos.obj.ssp.415* includes *perfagent.server.2.1.4.4*.
- Tools** Also part of PAIDE/6000 (*perfagent.tools*), but these are separate, stand-alone command-line tools. Their usage is described in *RS/6000 Performance Tools in Focus*, SG24-4989 and in the *AIX Version 4 Performance Tuning Guide*, SC23-2365.
- PTPE** Performance Toolbox Parallel Extensions (*ptpe.program*) adds additional agent support for SP-specific variables, options for hierarchical summarization on large SP systems, and an extension to the *Perspectives* interface for the Performance Monitor graphical user interface.

Note

In other IBM publications, the manager and agent aspects are also discussed, respectively, as client and server and/or as supplier and consumer of data statistics.

Part 1. The Performance Toolbox

Chapter 2. Manager - Basic Configuration

The basic configuration files of the Performance Toolbox (PTX) manager allow the user to go in and modify or tailor for their specific environment or needs. The installation process places these files in the /usr/lpp/perfmgr directory, with the exception of the XMperf file that is located in the /usr/lib/X11/app-defaults directory.

Before starting the file configuration, it is strongly recommended that the user copy the default configuration files into the /etc/perf or \$HOME directory so the original defaults can be used at any time.

In this chapter the following files were copied into the /etc/perf directory and will be the files that we use:

- Rsi.hosts (host configuration file)
- xmperf.cf (configuration file)
- xmperf.hlp (help file)

When the user goes into PTX and runs a specific command, PTX then searches for the files (except XMperf) in the following order and uses the first occurrence:

1. \$HOME (user's home directory)
2. /etc/perf
3. /usr/lpp/perfmgr (default install directory)

We copied the XMperf file into the /etc/perf directory for backup purposes only. XMperf is searched for in the /usr/lib/X11/app-defaults directory.

2.1 The XMperf Resource File

The XMperf resource file is used as a default display configuration for the xmperf command. This may be tailored to suit the users preference. The user has to be logged on as root (please make a backup copy first); then the colors or the font size can be tailored to suit the screen output.

2.1.1 The XMperf Font

The XMperf resource file uses the default font specified; however if there is no default in the file, it looks for a font from the following resources:

- graphfont
- FontList
- fontList
- Font
- font

If none of the above are found, then *Rom8* is the font used.

To change the size or style of a particular font in the console area, the user needs to go into /usr/lib/X11/app-defaults directory and then tailor the XMperf file. Below is an extract from the default XMperf file:

```
*GraphFont:          -ibm-block-medium-r-normal--15-100-100-100-c-70-iso8859-1
```

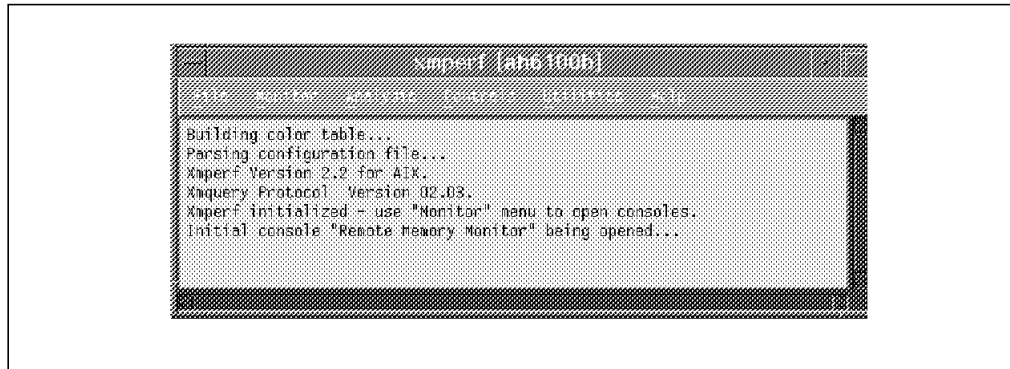


Figure 1. Main Menu - Default Font

There are two techniques to change your font:

1. Aliases - If you know the alias of a particular font, replace the default with your font alias as in the example below.

*GraphFont: Rom14.iso1

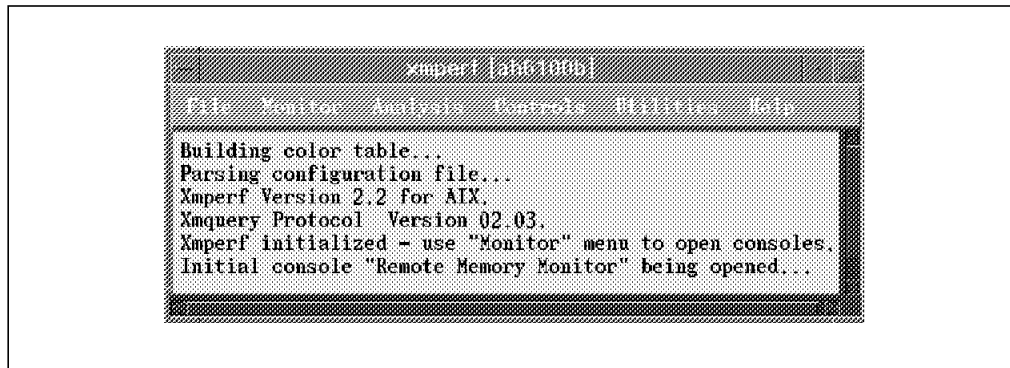


Figure 2. Main Menu - Alias Font

1. Full font string - If you know the full string of the font you prefer, replace the default as in the example below.

*GraphFont: -ibm-typewriter-bold-r-normal-iso9241-19-180-75-75-m-120-iso8859-1

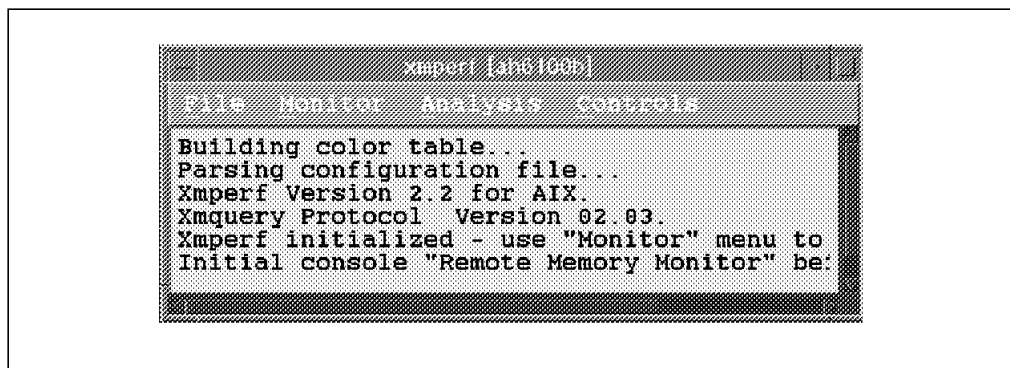


Figure 3. Main Menu - Full String Font

Note

You may use the `/usr/bin/X11/custom/bin/custom` command to see all available fonts on your system or use the `xset q` command to list the directories where all the system fonts exist.

It is recommended that you comment out the default font and copy in your desired font so that you may just uncomment the default if you wish to use it again.

Example:

```
#*GraphFont: -ibm-block-medium-r-normal--15-100-100-100-c-70-iso8859-1
*GraphFont: -ibm-typewriter-bold-r-normal-iso9241-19-180-75-75-m-120-iso8859-1
```

2.1.2 The XMperf Colors

Colors in the XMperf file relate to all the colors seen in the running of PTX; these may be tailored to suit any combination. All the user needs to do is select the desired color and option and type it over the destination color that you would like to replace.

These are the default colors available to you when using `xmperf`:

- black
- grey10
- grey20
- grey30
- grey40
- grey50
- grey60
- grey70
- grey80
- grey90
- ForestGreen
- goldenrod
- red
- MediumVioletRed
- LightSteelBlue
- SlateBlue
- green
- yellow
- BlueViolet
- SkyBlue
- pink
- GreenYellow
- SandyBrown
- orange
- plum
- MediumTurquoise
- LimeGreen
- chaki
- coral
- magenta
- cyan
- salmon

- sienna
- blue
- firebrick
- LightGrey
- SteelBlue
- grey45

The color selections below are from the *Color Selection* panel when adding or changing a specific value. The top part shows the default colors available as defined in XMperf; the bottom part shows the patterns as discussed in 2.1.4, “The XMperf Patterns” on page 11.

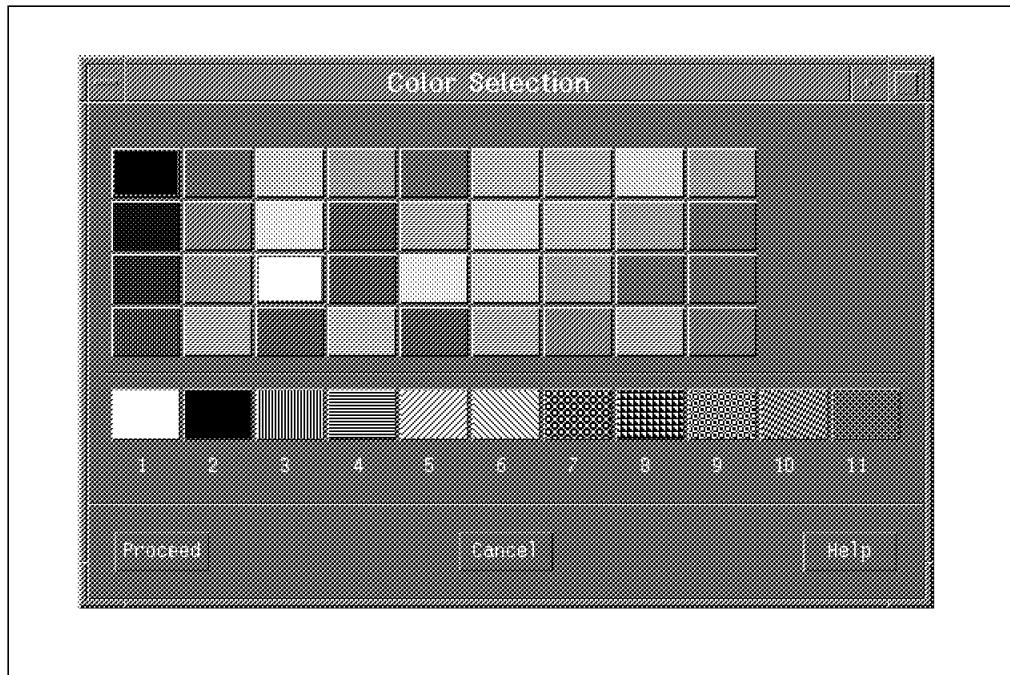


Figure 4. The xmperf Color Selection Panel

Note

To display all possible colors available to the user, you may run the command `/usr/lpp/X11/bin/showrgb | pg`, and you will see all the possible colors from which to choose.

2.1.3 The XMperf Shading

You may select the shading that you require for a particular instrument. Shading is in the range of 0 to 100, with 0 being dark or original color and 100 being light or the opposite color.

Example: You can see that the colors start from black (top left) and go down each column starting with grey10 and increasing by shades of 10 until grey90 is reached (2nd row, 3rd column) and then finishing with white and then green.

The shadings below are an extract from the color selection panel when adding or changing a specific value:

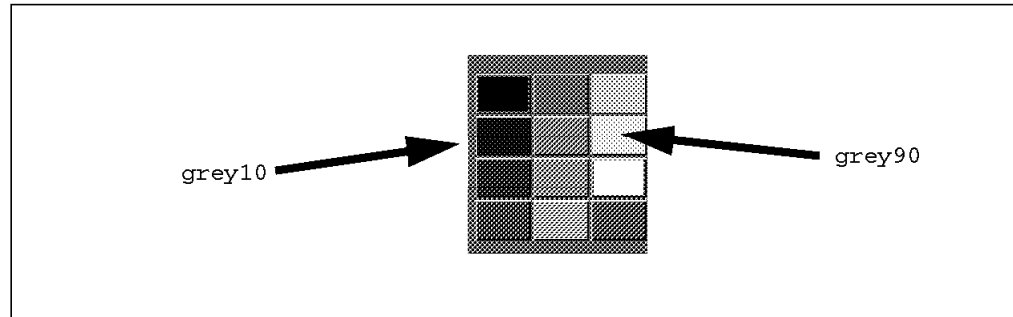


Figure 5. XMperf Shading

2.1.4 The XMperf Patterns

The patterns available for use in the xmperf.cf configuration file are as follows:

1. foreground
2. background
3. vertical
4. horizontal
5. slant_right
6. slant_left
7. plaid
8. triangles
9. wallpaper
10. zigzags
11. fabric

Example: The color white can be selected from the Color Selection panel:

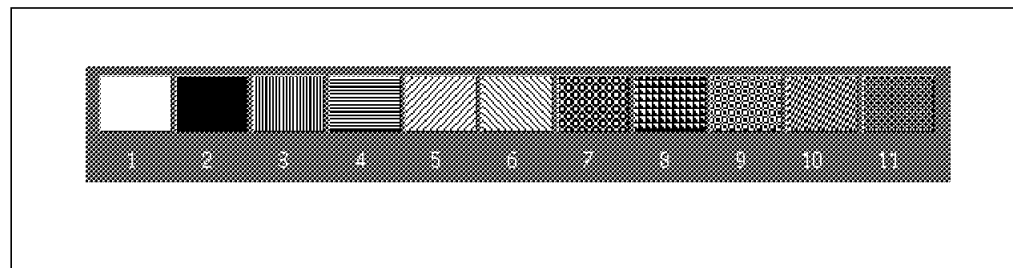


Figure 6. XMperf Patterns

Note

These patterns are described in the Help Index when you select **Color Selection (general)**.

2.1.5 The XMperf Customizing

This is an extract from the `/usr/lib/X11/app-defaults/XMperf` file that shows us the PTX functions and the system colors that they use. If you wish to change a specific field, all you have to do is type over the desired field color with the color of your choice.

```

*background:                grey70
*XMMenubar.background:      #cdb54d
*XmCascadeButton.background: #cdb54d
Console*XMMenubar.background: pink
Console*XmCascadeButton.background: pink
Console*XMMenubar.foreground: black
Console*XmCascadeButton.foreground: black
*XMProcMenubar.background:  blue
*XMProcList*XmCascadeButton.background: blue
*XMProcList*XmCascadeButton.foreground: turquoise
*XMSkelMenubar.background:  blue
*XMSkelList*XmCascadeButton.background: blue
*XMSkelList*XmCascadeButton.foreground: turquoise
*XMHostMenubar.background:  blue
*XMHostList*XmCascadeButton.background: blue
*XMHostList*XmCascadeButton.foreground: turquoise
*XMHelpMenubar.background:  ForestGreen
*XMHelp*XmCascadeButton.background: ForestGreen
*XMHelp*XmCascadeButton.foreground: yellow
*XMColorWindow.background:  black
*XMTabWindow.background:    grey70
*XMTabWindow.foreground:    black
#
#   Dialog Colors
#
*XMessage.background:       #eaeaad
*XMessage.foreground:       black
*XMOptions.background:      medium aquamarine
*XMOptions.foreground:      black
*XMDelete.background:       blue
*XMDelete.foreground:       yellow
*XMExit.background:         firebrick
*XMExit.foreground:         black
*XMChanged.background:      yellow
*XMChanged.foreground:      black
*XMHelp.background:         DarkGreen
*XMHelp.foreground:         MediumSpringGreen
*XMStop.background:         pink
*XMStop.foreground:         black
*XMMsgbox.background:       DarkGreen
*XMMsgbox.foreground:       light grey
#
#   Light Colors
#
*XMLight.background:        #729fff
*XMLight.foreground:        black
#
#   Digital Clock Colors
#
*XMSeekTime.foreground:     red
*XMSeekTime.background:     black
*XMPlayTime.foreground:     yellow
*XMPlayTime.background:     black
#
#           Default Value Colors
#
*ValueColor1:ForestGreen
*ValueColor2:goldenrod
*ValueColor3:red

```

```
*ValueColor4:MediumVioletRed
*ValueColor5:LightSteelBlue
*ValueColor6:SlateBlue
*ValueColor7:green
*ValueColor8:yellow
*ValueColor9:BlueViolet
*ValueColor10:SkyBlue
*ValueColor11:pink
*ValueColor12:GreenYellow
*ValueColor13:SandyBrown
*ValueColor14:orange
*ValueColor15:plum
*ValueColor16:MediumTurquoise
*ValueColor17:LimeGreen
*ValueColor18:khaki
*ValueColor19:coral
*ValueColor20:magenta
*ValueColor21:cyan
*ValueColor22:salmon
*ValueColor23:sienna
*ValueColor24:blue
```

The instrument graphs use the default color values in ascending order.

Example: The first graph color in an instrument is ForestGreen; the next will be goldenrod, proceeding in ascending order until blue is reached.

2.1.6 XMperf LegendWidth

If you require your graph to start up with a specific value for the LegendWidth, insert the following into the XMperf resource file.

```
*LegendWidth: xx
```

Where xx is the width specified. Its range is from 8 to 32. Refer to 2.3.1, “The xmperf Options” on page 15, for a detailed example of the LegendWidth.

2.2 The Rsi.hosts Host File

The Rsi.hosts file is used by PTX to recognize specific IP addresses, hostnames or subnets of your network. The default is to broadcast to the entire network that the machine is on, thereby causing the machine to have poor initial response time while locating each individual machine that is set up to run the xmservd daemon. To overcome this problem, it is advised that you broadcast to a specific IP address, hostname or subnet. To make these changes, edit your Rsi.hosts file as in the example below.

This is an excerpt from the default Rsi.hosts file.

```
# To suppress broadcast on all LAN interfaces, uncomment the
# following line.
#nobroadcast
# The following lines illustrate broadcasting to hosts on specific
# subnets. To activate, uncomment the lines.
#129.21.15.255
#9.55.255.255
#129.5.47.255
# The following lines illustrate broadcasting to specific hosts.
# To activate, uncomment the lines.
```

```
#birte  
#bongo.austin.ibm.com
```

Below is the configuration of specific IP addresses, hostnames and subnets that a user would like to see.

Example:

```
#To suppress broadcast on all LAN interfaces, uncomment the  
# following line.  
nobroadcast # Disable the Broadcast option  
# The following lines illustrate broadcasting to hosts on specific  
# subnets. To activate, uncomment the lines.  
9.3.1.255 # Using Subnet  
# The following lines illustrate broadcasting to specific hosts.  
# To activate, uncomment the lines.  
9.3.1.71 # Using IP Address  
ah6100a # Using Hostnames
```

Note

After making the desired changes to the Rsi.hosts file, you may encounter problems in finding the hostname(s) specified. To overcome this, you need to restart xmperf. Usually, the second around time, it works. You may also use the -r option in the startup of xmperf and specify a longer initialization time. Please refer to 2.3.1, "The xmperf Options" on page 15, for more details.

2.3 The xmperf.cf Configuration File

The xmperf program is a graphical performance monitor, that uses the xmperf.cf file for its configuration. You may use xmperf for monitoring a local machine or specific machines on your network that have the agent properly installed.

Each xmperf environment can be kept in a separate configuration file. The default is the xmperf.cf file, which is searched for in the following order: \$HOME directory, /etc/perf directory and last /usr/lpp/perfmgr directory. The user may also use another configuration file that is specified by the -o option while starting up xmperf.

The xmperf.cf configuration file is divided into five sections. Below is a brief description of each part:

1. ctrl - defines the tools used. This is explained in 2.3.7.3, "Controls - Menu Bar Pull-down" on page 41.
2. util - defines the utilities pull-down menu. This is explained in 2.3.7.1, "Utilities - Menu Bar Pull-down" on page 34.
3. analy - defines the analysis pull-down menu. This is explained in 2.3.7.2, "Analysis - Menu Bar Pull-down" on page 38.
4. proc - defines the process selection within the controls pull-down. This is explained in 2.3.7.4, "Process Controls" on page 43.
5. monitor - defines the monitor pull-down menu used. This is explained in 2.3.2, "The xmperf Console" on page 17.

Refer to *Performance Toolbox for AIX: Guide and Reference*, SC23-2625, *Appendix B* for detailed information about the individual sections.

Note

If you make any changes and for some reason this does not work, the error will be recorded in the `xmperf.log` file located in the `$HOME` directory.

2.3.1 The xmperf Options

There are several options that you may wish to use with the `xmperf` command. Below is a list of the more commonly used ones. Refer to *Performance Toolbox for AIX: Guide and Reference*, SC23-2625, *Chapter 3* for more detailed information on all options used with `xmperf`.

-v Verbose. This option prints the configuration lines as they are processed into the `$HOME/xmperf.log` file. It is very useful. If you make changes to the `xmperf.cf` file and encounter errors, you can reference them through the `xmperf.log` file. This should only be used when testing a new configuration.

Example: `xmperf -v`

-w LegendWidth. The option is used to vary the width of the legend in the graphs that the user will use. The default is 12. The user can use values in the range of 8 to 32 depending on the length of the legend titles. Legend is the name of the statistic PTX is using, and LegendWidth is the size of the field that the statistic description is using. In the examples below, User, Kernel, Wait, and Idle are the legend, and the field width is shown to both extremes.

Example: `xmperf -w 8`

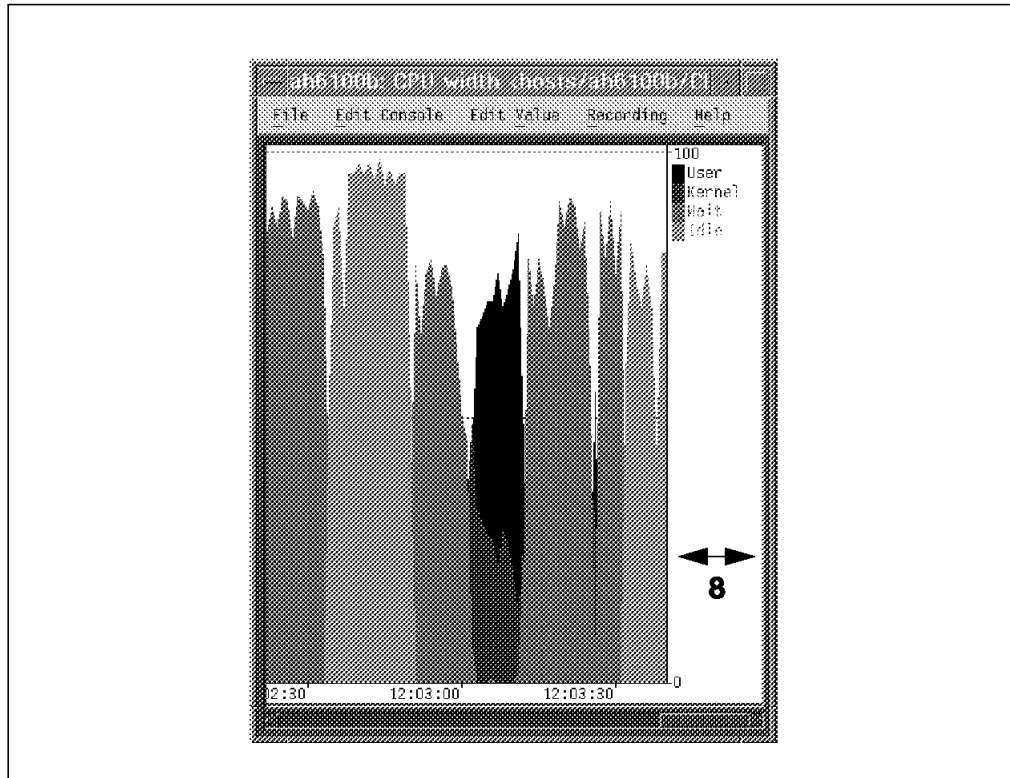


Figure 7. LegendWidth 8

Example: `xmperv -w 32`

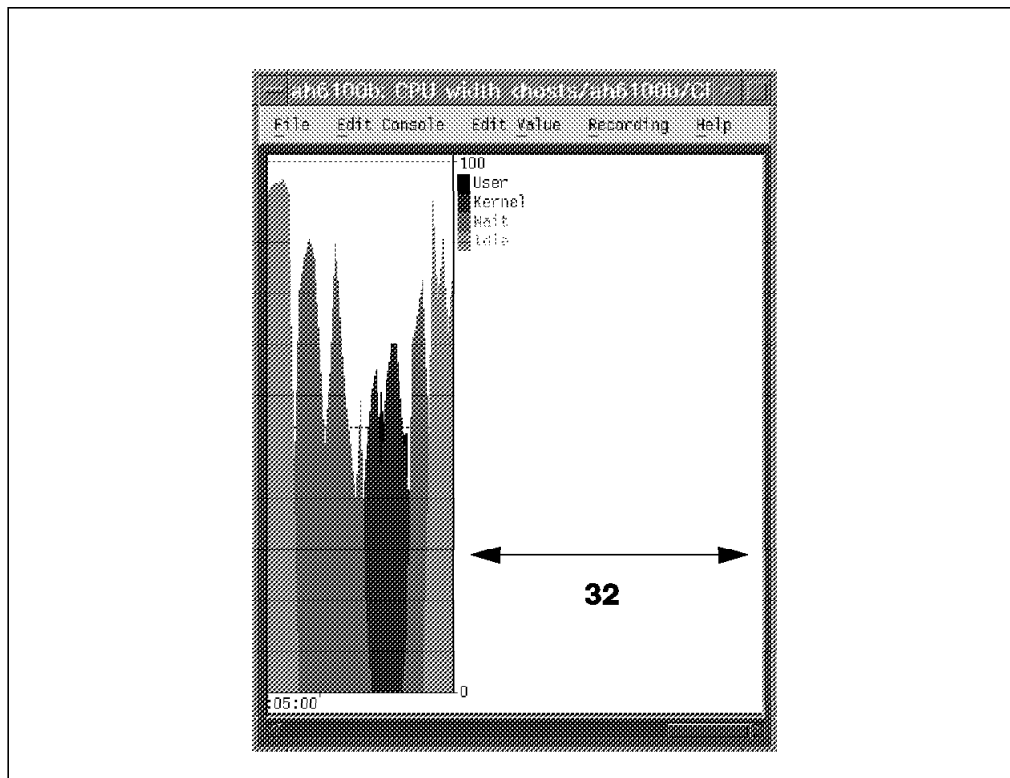


Figure 8. LegendWidth 32

Note

You may wish to set your default LegendWidth to a specified number. If so, please refer to 2.1.6, "XMperf LegendWidth" on page 13.

- o ConfigFile. This option is used to specify a configuration file to use. The default is to use the first available xmperf.cf file found in the \$HOME directory or in the /etc/perf directory or in the /usr/lpp/perfagent directory, in this order. If you wish to have different setups for different environments, this is the option that you would choose.

Example: xmperf -o xmperf.remote

- h Host Monitor. This option is used when you wish to monitor a remote host and use it as the local host. The -h option is a handy option you may wish to use from your machine to monitor a server or network machine.

Example: xmperf -h ah6100c

- r Specifies the timeout in milliseconds for the remote hosts. The range is from 5 to 10000 milliseconds, and the default is 100 milliseconds.

Example: xmperf -r 1000

Note

You may use more than one option in the startup of xmperf.

Example: xmperf -v -w 8 -h ah6100a

2.3.2 The xmperf Console

The main feature of the xmperf program is the use of consoles. Each console can be tailored to the user's specifications. A console consists of a number of instruments depending on the user's environment requirements.

2.3.2.1 The Default Console

The default console for xmperf is *Mini Monitor*, which will automatically come up on the display when xmperf is started, if xmperf.cf has not been modified. It is specified in the monitor section of the xmperf.cf configuration file as seen below:

```
monitor.Mini Monitor.default
```

Below is the output of the default Mini Monitor console:

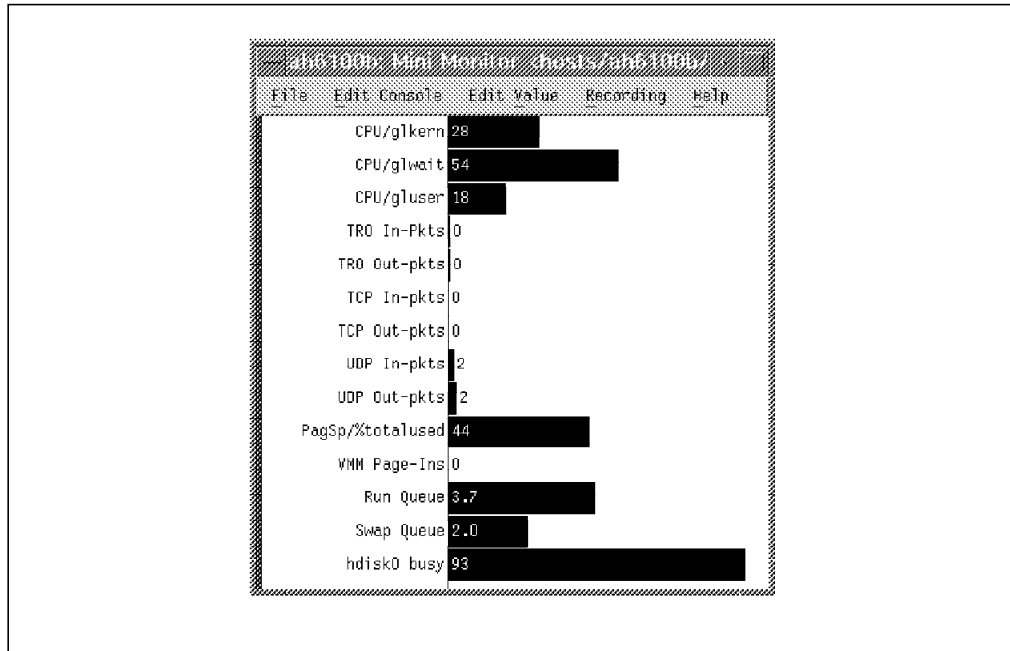


Figure 9. Default Mini Monitor

With the default console setup in `xmperf.cf`, we have three main options:

1. No default setup - All you need to do is comment out the current default setup and `xmperf` will start with no console coming up.

Example:

```
#monitor.Mini Monitor.default
```

2. One default setup - Add the default setup that you would like to start with (for example: Local System Monitor) and make sure the Mini Monitor has been commented out with the `#`, or leave as the original default and Mini Monitor will start.

Example:

```
#monitor.Mini Monitor.default
monitor.Local System Monitor.default
```

3. More than one default setup - To start multiple consoles up, all you need to do is specify them in the `xmperf.cf` file as stated below.

Example:

```
monitor.Mini Monitor.default
monitor.Local System Monitor.default
```

Note

The general form of the default monitor setup is:
`monitor.(console name).default`

2.3.3 Console Keywords

Let's first explain the keywords that are unique to the console:

| | |
|---------------|---|
| width | The width of the console |
| height | The height of the console |
| x | The position of the console on the x axis |
| y | The position of the console on the y axis |

The important keywords for each instrument:

| | |
|-------------------|---|
| left | Left position of instrument within console in percent |
| top | Top position of instrument within console in percent |
| right | Right position of instrument within console in percent |
| bottom | Bottom position of instrument within console in percent |
| background | Background color of instrument |
| foreground | Foreground color of instrument |

The important keywords for a value within an instrument:

| | |
|---------------------|--|
| input.n | The name of the statistic |
| color.n | The color of the statistic |
| range.n | The range of the statistic |
| thresh.n | The threshold of the statistic |
| label.n | The name being displayed in the instrument |
| descending.n | If this appears, it means that the light is turned on if the value is below the threshold. |

2.3.4 Local Instruments

Consoles consist of one or many instruments. These local instruments are used to monitor resources on a local machine. A good example of the use of a local instrument is to analyze the CPU usage while using the machine. Below is a detailed example of this configuration in the xmperf.cf file.

The time and date of when the console was set up:

```
# monitor.Local CPU Monitor., saved at Tue Jul 15 15:00:57 1997
```

The next section defines the startup size of the console and its location on the screen. We have to choose from a maximum width of 1268 and height of 989 pixels. This particular instrument is located at the top left of the screen (x: = 0 and y: = 0), with the size being 600 pixels wide and 989 pixels in height. The 1 after the name Local CPU Monitor is somewhat of a place holder for the console.

```
monitor.Local CPU Monitor.1.width:    600
monitor.Local CPU Monitor.1.height:   989
monitor.Local CPU Monitor.1.x:       0
monitor.Local CPU Monitor.1.y:       0
```

Next we define the first instrument within the console. The sizes for top to bottom and left to right are broken up into percentages.

Starts at the top of the screen:

```
monitor.Local CPU Monitor.1.top:      1
```

Start at the left side of the screen:

```
monitor.Local CPU Monitor.1.left:     1
```

Goes 99 percent to the right of the screen:

```
monitor.Local CPU Monitor.1.right: 99
```

Goes 25 percent down the screen:

```
monitor.Local CPU Monitor.1.bottom: 25
```

Shifting is only meaningful for recording graphs. It determines the number of pixels the graph should move after each reading. The default is 4, but you may use the range from 1 to 20:

```
monitor.Local CPU Monitor.1.shift: 4
```

Spacing is meaningful for bar and state graphs. It defines the number of pixels separating each bar:

```
monitor.Local CPU Monitor.1.space: 2
```

History is meaningful for recording graphs only. It defines the number of statistic readings to keep. The range is 50 to 5000, with a default of 500 readings.

```
monitor.Local CPU Monitor.1.history: 500
```

The interval between observations in milliseconds, in the range of 0.2 seconds to 30 minutes. The default is 5 seconds.

```
monitor.Local CPU Monitor.1.interval: 5000
```

The background color of the instrument:

```
monitor.Local CPU Monitor.1.background: white
```

The foreground color of the instrument:

```
monitor.Local CPU Monitor.1.foreground: black
```

For the style of the graph, please refer to 2.3.6.3, "Recording Graphs" on page 30, for recording styles and to 2.3.6.4, "State Graphs" on page 31, for state graph styles:

```
monitor.Local CPU Monitor.1.style: area
```

The particular statistic being monitored:

```
monitor.Local CPU Monitor.1.input.1: CPU/gluser
```

The color of the statistic graph:

```
monitor.Local CPU Monitor.1.color.1: black
```

The range of the statistic display:

```
monitor.Local CPU Monitor.1.range.1: 0-100
```

Instrument two is defined next. The only differences from the first instrument are in the initial positioning. The top is at 26 percent and the bottom is at 50 percent. The number 2 is the instrument number, and glkern is the statistic to monitor.

```
monitor.Local CPU Monitor.2.left:      1
monitor.Local CPU Monitor.2.top:       26
monitor.Local CPU Monitor.2.right:     99
monitor.Local CPU Monitor.2.bottom:    50
monitor.Local CPU Monitor.2.shift:     4
monitor.Local CPU Monitor.2.space:     2
monitor.Local CPU Monitor.2.history:   500
monitor.Local CPU Monitor.2.interval:  5000
monitor.Local CPU Monitor.2.background: white
monitor.Local CPU Monitor.2.foreground: black
monitor.Local CPU Monitor.2.style:     area
monitor.Local CPU Monitor.2.input.2:   CPU/glkern
monitor.Local CPU Monitor.2.color.2:   black
monitor.Local CPU Monitor.2.range.2:   0-100
```

Instrument three is the same as above but with the top location at 51 percent, the bottom location at 75 percent, assigning the instrument number to 3 and using `glwait` as the statistic to monitor.

```
monitor.Local CPU Monitor.3.left:      1
monitor.Local CPU Monitor.3.top:       51
monitor.Local CPU Monitor.3.right:     99
monitor.Local CPU Monitor.3.bottom:    75
monitor.Local CPU Monitor.3.shift:     4
monitor.Local CPU Monitor.3.space:     2
monitor.Local CPU Monitor.3.history:   500
monitor.Local CPU Monitor.3.interval:  5000
monitor.Local CPU Monitor.3.background: white
monitor.Local CPU Monitor.3.foreground: black
monitor.Local CPU Monitor.3.style:     area
monitor.Local CPU Monitor.3.input.3:   CPU/glwait
monitor.Local CPU Monitor.3.color.3:   black
monitor.Local CPU Monitor.3.range.3:   0-100
```

Instrument four is the same as above but with the top location at 76 percent, the bottom location at 99 percent, assigning the instrument number to 4 and using `glidle` as the statistic to monitor.

```
monitor.Local CPU Monitor.4.left:      1
monitor.Local CPU Monitor.4.top:       76
monitor.Local CPU Monitor.4.right:     99
monitor.Local CPU Monitor.4.bottom:    99
monitor.Local CPU Monitor.4.shift:     4
monitor.Local CPU Monitor.4.space:     2
monitor.Local CPU Monitor.4.history:   500
monitor.Local CPU Monitor.4.interval:  5000
monitor.Local CPU Monitor.4.background: white
monitor.Local CPU Monitor.4.foreground: black
monitor.Local CPU Monitor.4.style:     area
monitor.Local CPU Monitor.4.input.4:   CPU/glidle
monitor.Local CPU Monitor.4.color.4:   black
monitor.Local CPU Monitor.4.range.4:   0-100
```

After inserting the above code into your `xmperf.cf` file, you will now see *Local CPU Monitor* appear in the Monitor pull-down menu:

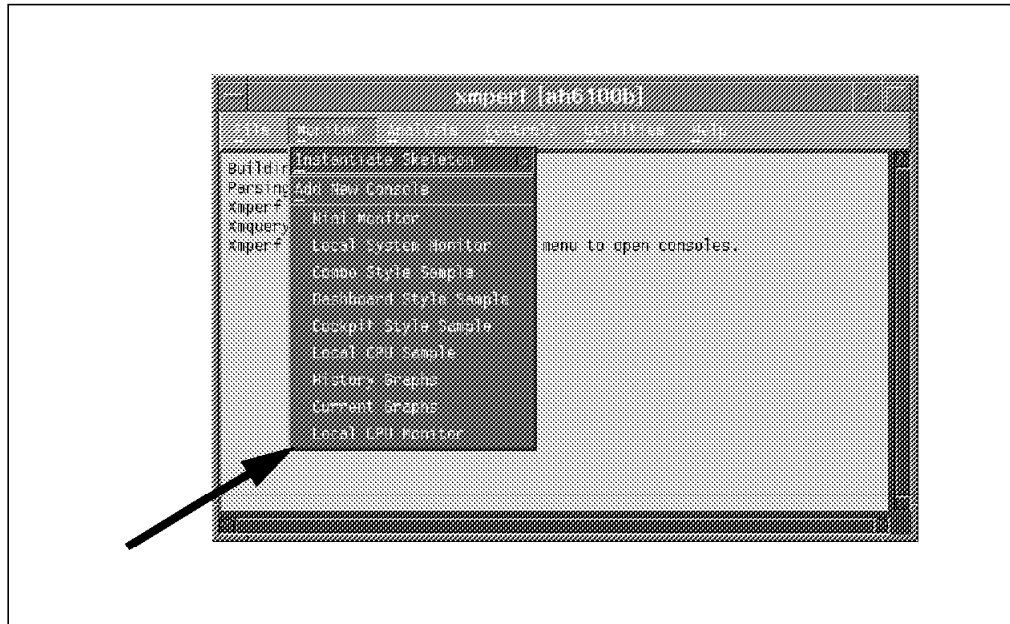


Figure 10. Local CPU Monitor Pull-down

Below is the output of the Local CPU Monitor configuration as defined in the above code:

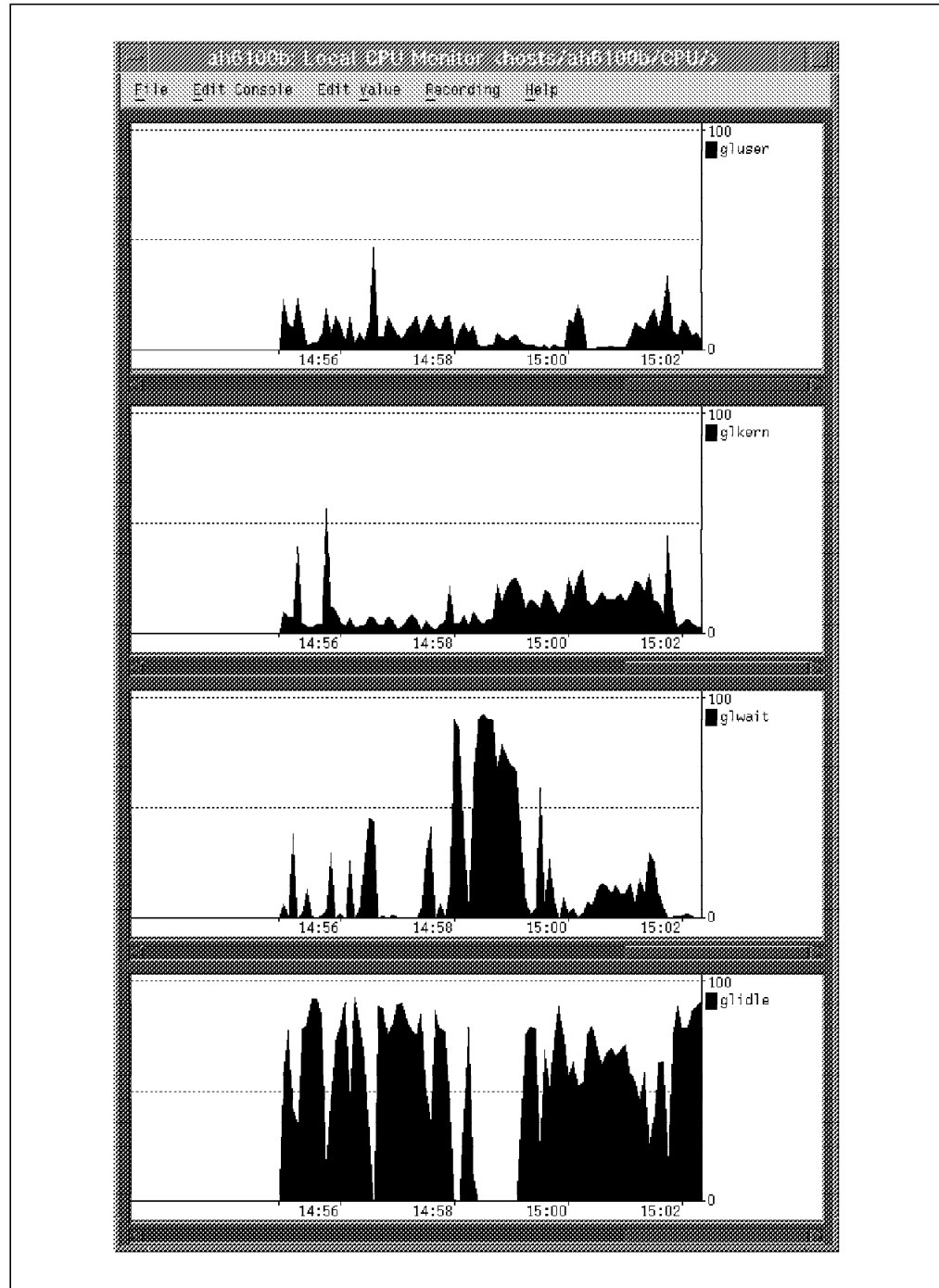


Figure 11. Local CPU Monitor

2.3.5 Remote Instruments

PTX gives you the opportunity to analyze remote systems from the local machine that you are logged on to. We have selected the following to highlight the changes you can make to display the remote system:

This segment locates the instrument of %free memory to the top (1) and left (1) of the screen and allows the dimensions to the middle of the screen with bottom (50) and right (50):

```
monitor.Remote Memory Monitor.1.left: 1
monitor.Remote Memory Monitor.1.top: 1
monitor.Remote Memory Monitor.1.right: 50
monitor.Remote Memory Monitor.1.bottom:50
monitor.Remote Memory Monitor.1.input.1:          hosts/itsosmp/Mem/Real/%free
```

The segment %pinned memory graph starts half way across the screen at the top (1) and left (51), ending to the far right (99) and half way down the screen bottom (50).

```
monitor.Remote Memory Monitor.2.left: 51
monitor.Remote Memory Monitor.2.top: 1
monitor.Remote Memory Monitor.2.right: 99
monitor.Remote Memory Monitor.2.bottom:50
monitor.Remote Memory Monitor.2.input.1:          hosts/itsosmp/Mem/Real/%pinned
```

The segment %comp starts on the left (1) half way down (51) and finishes in the middle (50) at the bottom (99).

```
monitor.Remote Memory Monitor.3.left: 1
monitor.Remote Memory Monitor.3.top: 51
monitor.Remote Memory Monitor.3.right: 50
monitor.Remote Memory Monitor.3.bottom:99
monitor.Remote Memory Monitor.3.input.1:          hosts/itsosmp/Mem/Real/%comp
```

The segment %noncomp starts in the middle of the screen, left (51) and top (51), then goes to the far right (99) and bottom (99) of the screen.

```
monitor.Remote Memory Monitor.4.left: 51
monitor.Remote Memory Monitor.4.top: 51
monitor.Remote Memory Monitor.4.right: 99
monitor.Remote Memory Monitor.4.bottom:99
monitor.Remote Memory Monitor.4.input.1:          hosts/itsosmp/Mem/Real/%noncomp
```

If you now select the **Monitor** pull-down option in xmperv, you will see the **Remote Memory Monitor** option as below:

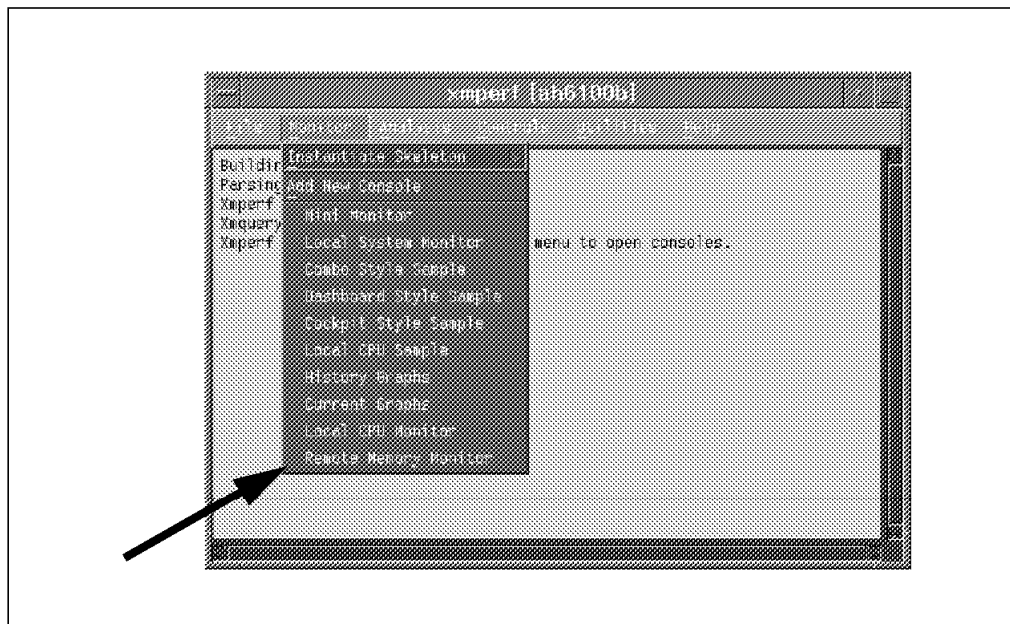


Figure 12. Remote Memory Monitor Pull-down

After you select the **Remote Memory Monitor** from the Monitor pull-down menu, it provides the following display:

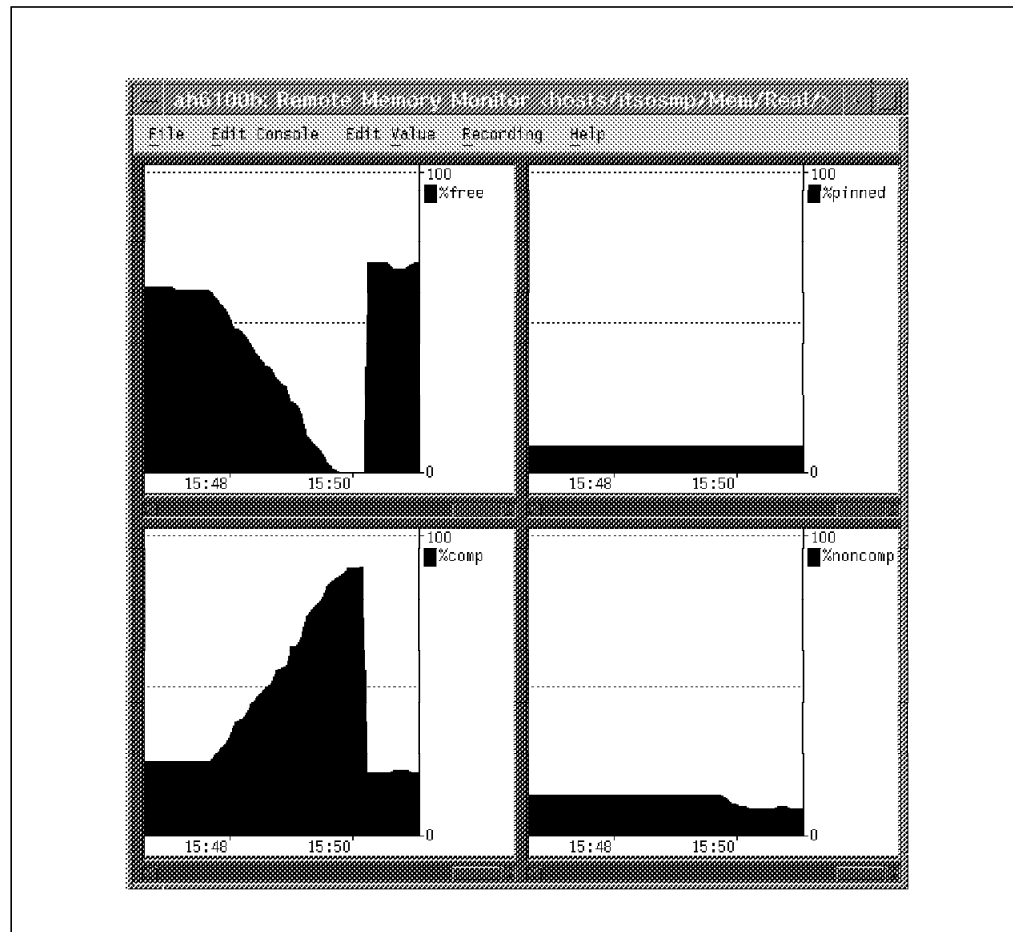


Figure 13. Remote Memory Monitor

2.3.6 Skeleton Instruments

The skeleton instruments option allows the user to select specific hardware on the local machine or hosts on the network. The skeleton uses a wildcard character which then prompts the user with a screen and asks the user to select the specific hardware or hosts to monitor. In this example we look into file systems on the local machine as well as those on the hosts.

2.3.6.1 Local File Systems

PTX allows the user to monitor specific file systems. The following segment of code was inserted into the `xmperf.cf` file and produced the output seen below.

```
monitor.Local File Systems.1.style: level
monitor.Local File Systems.1.all.2: FS/rootvg/*/%totfree
monitor.Local File Systems.1.color.2: black
monitor.Local File Systems.1.range.2: 0-100
```

Note

The `.all` option specifies to PTX to run all the statistics in one instrument.

After inserting the complete code into the xmperv.cf file, Local File Systems (2) will appear in the Instantiate Skeleton (1) option in the Monitor menu pull-down area:

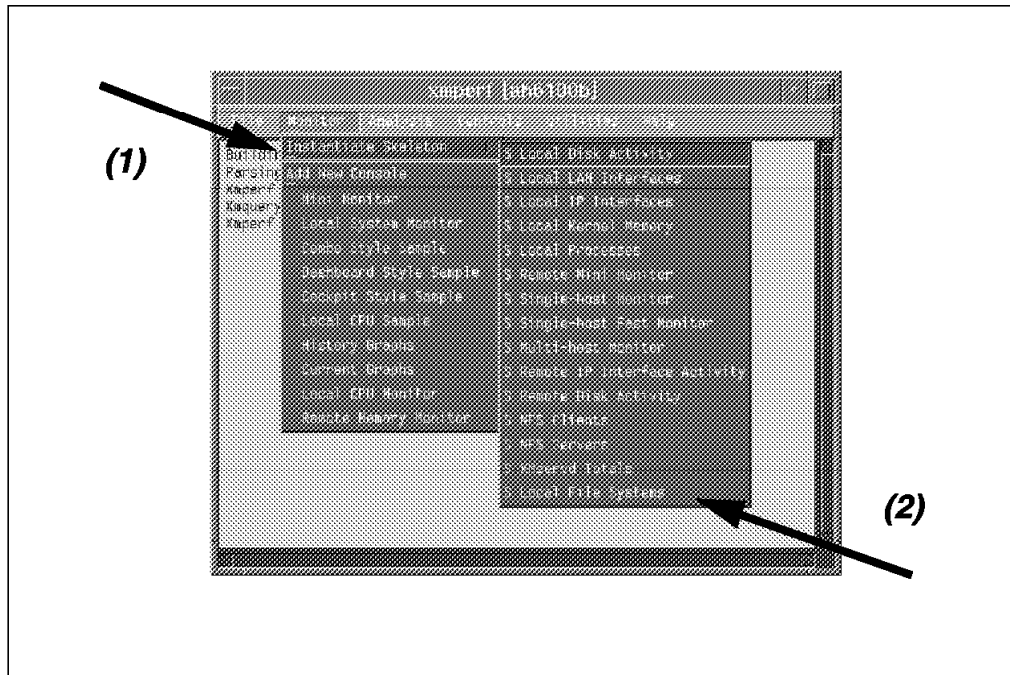


Figure 14. Local File Systems Pull-down

After you select the **Local File Systems** monitor in the skeleton, you are prompted with a similar screen to the one shown in Figure 15. Let's select file systems hd4 (/), hd2 (/usr), hd9var (/var), hd3 (/tmp), and hd1 (/home):

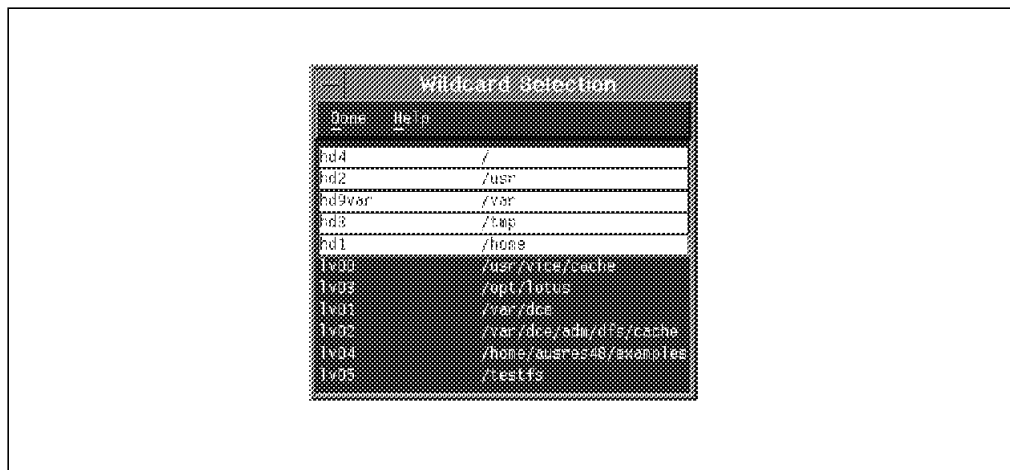


Figure 15. Selected Local File Systems

After selecting the file systems, click on the **Done** and then on **Accept Selection**, and the following will appear:

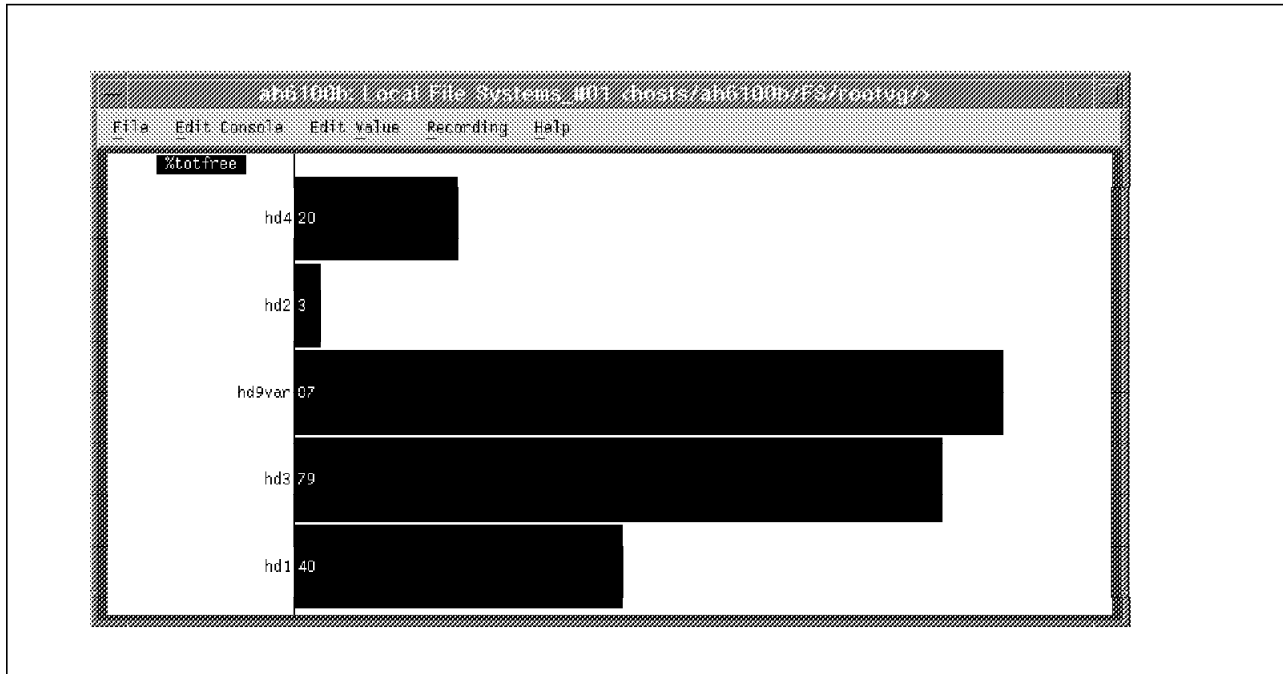


Figure 16. Local File System Console

2.3.6.2 Host File Systems

PTX also allows the user to monitor specific remote file systems. The following segments of code were inserted into the xmperf.cf file and produced the output as seen in Figure 19 on page 29.

```

monitor.Remote File Systems.1.each.2: hosts/*/FS/rootvg/hd2/%totfree
monitor.Remote File Systems.1.label.2: /usr (Free Space)
monitor.Remote File Systems.1.each.3: hosts/*/FS/rootvg/hd2/%totused
monitor.Remote File Systems.1.label.3: /usr (Used Space)
monitor.Remote File Systems.2.each.2: hosts/*/FS/rootvg/hd9var/%totfree
monitor.Remote File Systems.2.label.2: /var (Free Space)
monitor.Remote File Systems.2.each.3: hosts/*/FS/rootvg/hd9var/%totused
monitor.Remote File Systems.2.label.3: /var (Used Space)
monitor.Remote File Systems.3.each.2: hosts/*/FS/rootvg/hd3/%totfree
monitor.Remote File Systems.3.label.2: /tmp (Free Space)
monitor.Remote File Systems.3.each.3: hosts/*/FS/rootvg/hd3/%totused
monitor.Remote File Systems.3.label.3: /tmp (Used Space)
monitor.Remote File Systems.4.each.2: hosts/*/FS/rootvg/hd1/%totfree
monitor.Remote File Systems.4.label.2: /home (Free Space)
monitor.Remote File Systems.4.each.3: hosts/*/FS/rootvg/hd1/%totused
monitor.Remote File Systems.4.label.3: /home (Used Space)
monitor.Remote File Systems.5.each.1: hosts/*/FS/rootvg/hd2/%nodesfree
monitor.Remote File Systems.5.label.1: /usr (Free Nodes)
monitor.Remote File Systems.5.each.2: hosts/*/FS/rootvg/hd2/%nodesused
monitor.Remote File Systems.5.label.2: /usr (Used Nodes)
monitor.Remote File Systems.6.each.1: hosts/*/FS/rootvg/hd9var/%nodesfree
monitor.Remote File Systems.6.label.1: /var (Free Nodes)
monitor.Remote File Systems.6.each.2: hosts/*/FS/rootvg/hd9var/%nodesused
monitor.Remote File Systems.6.label.2: /var (Used Nodes)
monitor.Remote File Systems.7.each.1: hosts/*/FS/rootvg/hd3/%nodesfree
monitor.Remote File Systems.7.label.1: /tmp (Free Nodes)
monitor.Remote File Systems.7.each.2: hosts/*/FS/rootvg/hd3/%nodesused
monitor.Remote File Systems.7.label.2: /tmp (Used Nodes)
monitor.Remote File Systems.8.each.1: hosts/*/FS/rootvg/hd1/%nodesfree

```

```
monitor.Remote File Systems.8.label.1: /home (Free Nodes)
monitor.Remote File Systems.8.each.2:  hosts/*/FS/rootvg/hd1/%nodesused
monitor.Remote File Systems.8.label.2: /home (Used Nodes)
```

Note

The `.each` option specifies to PTX to run each statistic in separate instruments.

After inserting the complete code into the `xmperf.cf` file, Remote File Systems appears in the Instantiate Skeleton section of the Monitor pull-down:

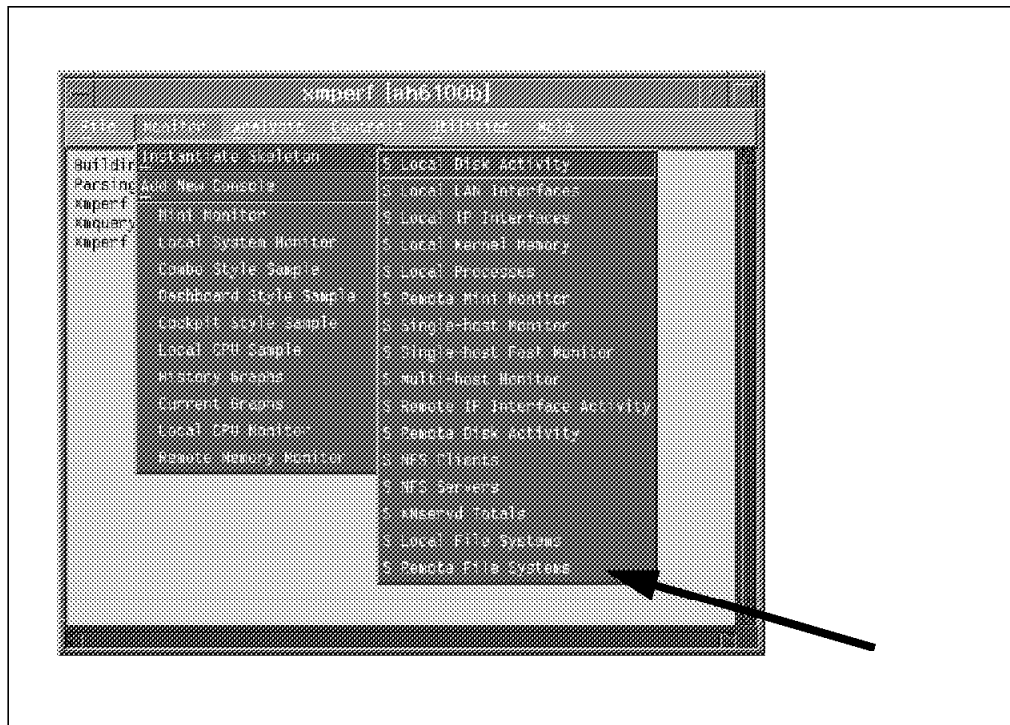


Figure 17. Remote File System Pull-down

When you select the **Remote File Systems** monitor in the Instantiate Skeleton menu, you are prompted with a similar screen to that seen in Figure 18 on page 29. Let's select two remote hosts, **ah6100c** and **itsosmp**:

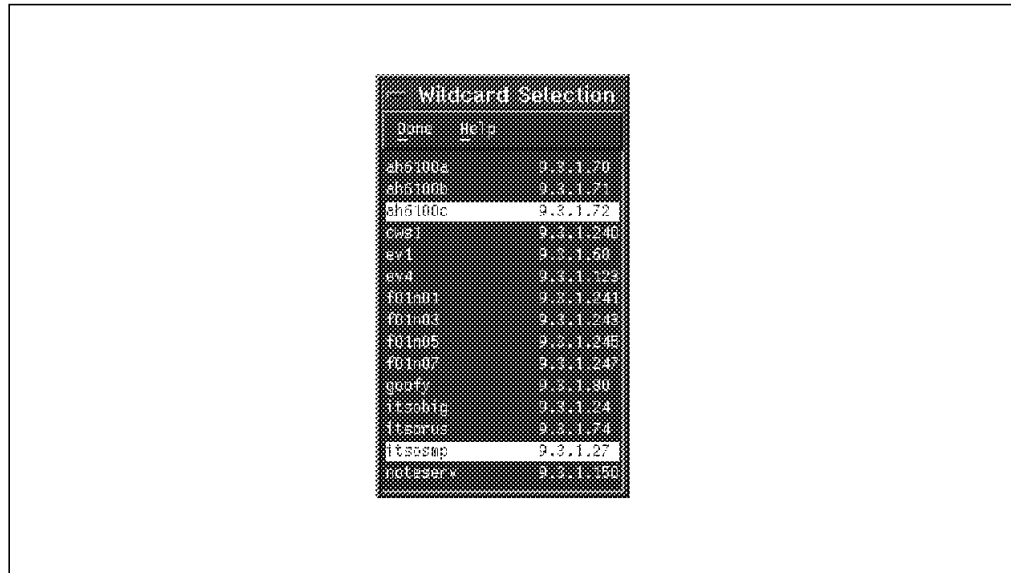


Figure 18. Host Selection - Remote File Systems

After selecting these hosts, we then pull-down the Done menu. Select **Accept Selection**, and the following screen will appear:

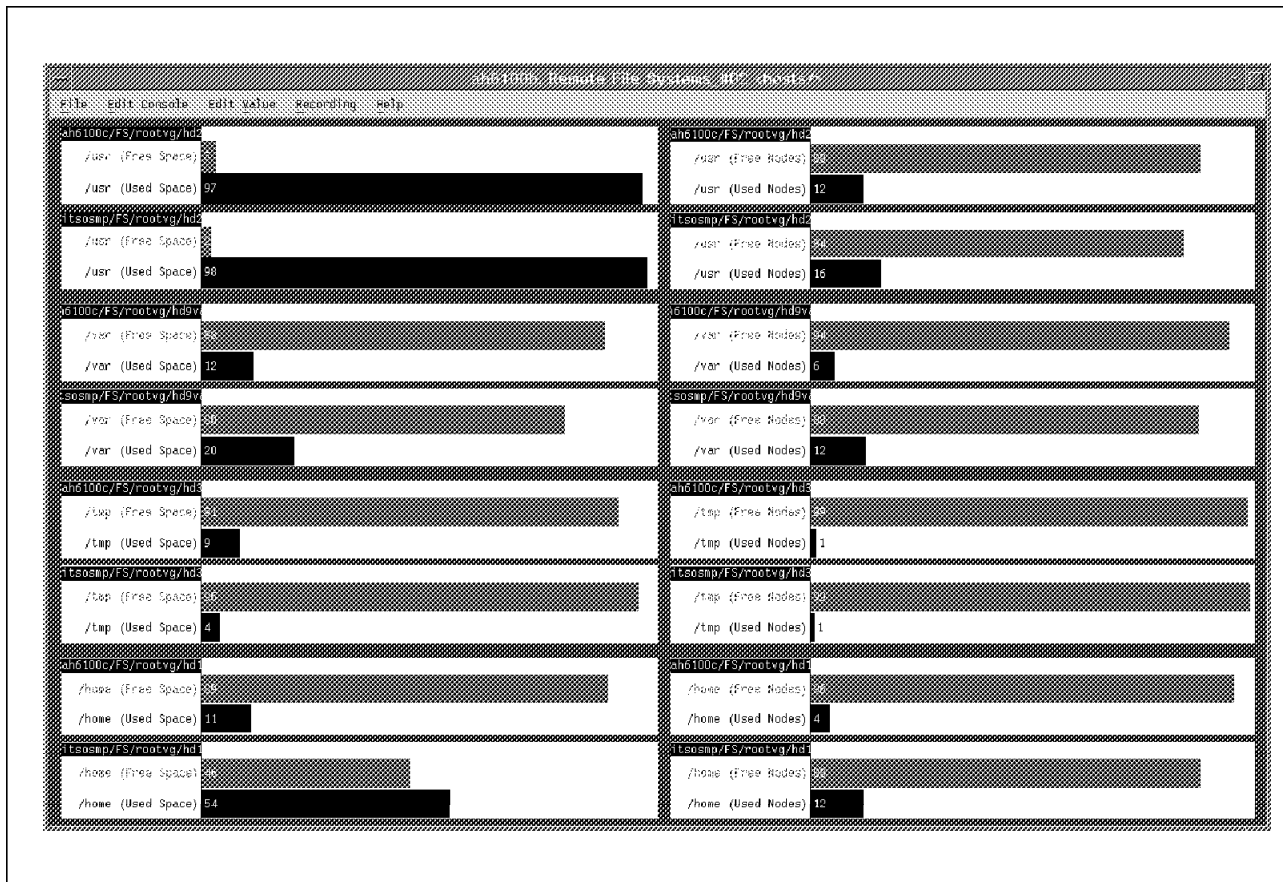


Figure 19. Remote File System Monitor

2.3.6.3 Recording Graphs

The following five styles were used to record events over a period of time to allow the user to analyze the system. These styles are explained in Figure 21 on page 31.

```
monitor.History Graphs.1.style:      line
monitor.History Graphs.2.style:      area
monitor.History Graphs.3.style:      skyline
monitor.History Graphs.4.style:      bar
monitor.History Graphs.5.style:      area
```

The stacked option, as seen in Figure 21 on page 31, adds the values up and then outputs the result to the screen.

```
monitor.History Graphs.5.stacked:    true
```

After inserting the complete code into your xmperf.cf file, you now see History Graphs appear in the Monitor pull-down menu:

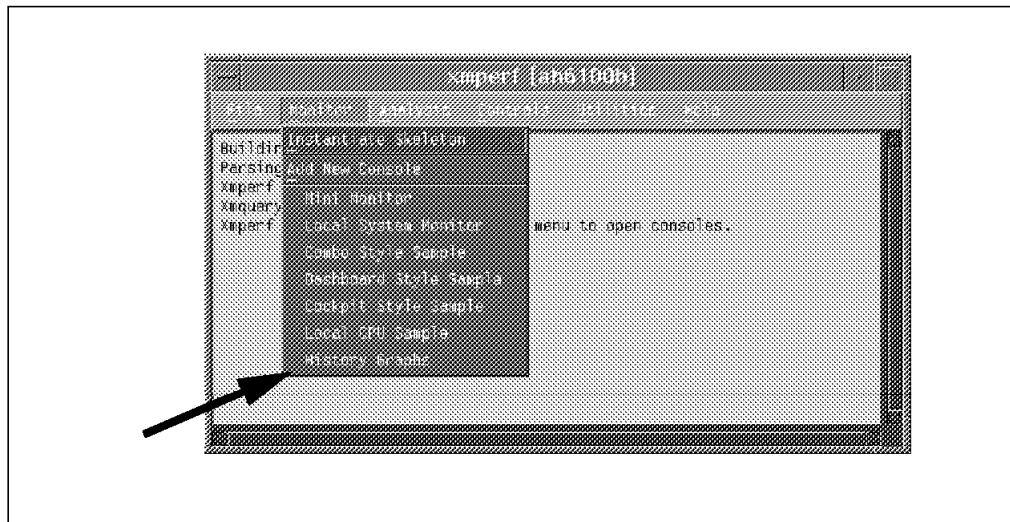


Figure 20. History Graphs Pull-down

After selecting **History Graphs**, the following appears on your screen:

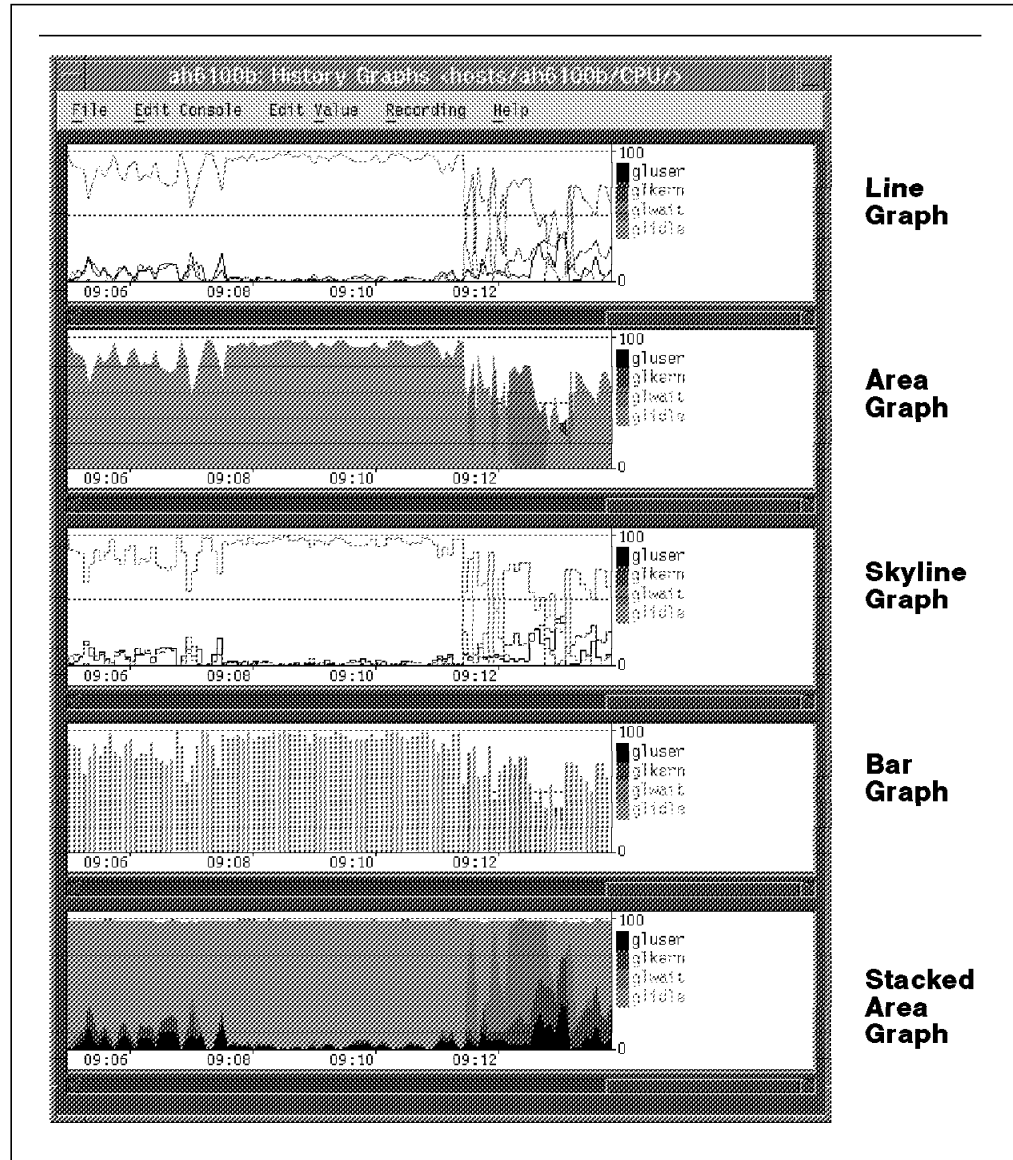


Figure 21. History Graphs

2.3.6.4 State Graphs

The following four styles are used for seeing the current status of the system. They are explained in Figure 23 on page 33.

```
monitor.Current Graphs.1.style: level
monitor.Current Graphs.2.style: light
monitor.Current Graphs.3.style: piechart
monitor.Current Graphs.4.style: meter
```

After inserting the complete code into your `xmperf.cf` file, you now see Current Graphs appear in the Monitor pull-down menu:

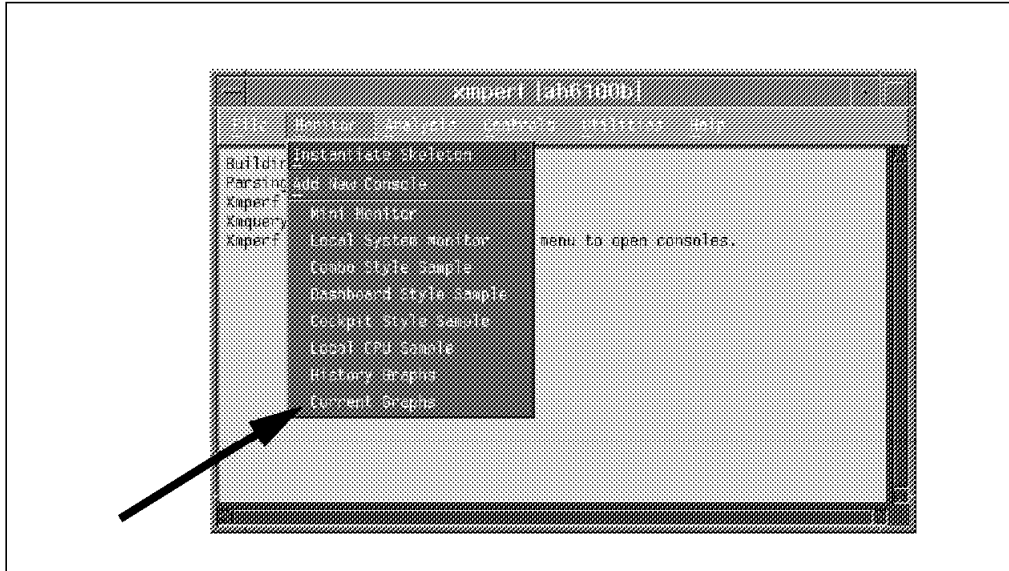


Figure 22. Current Graphs Pull-down

After selecting **Current Graphs**, the following appears on your screen:

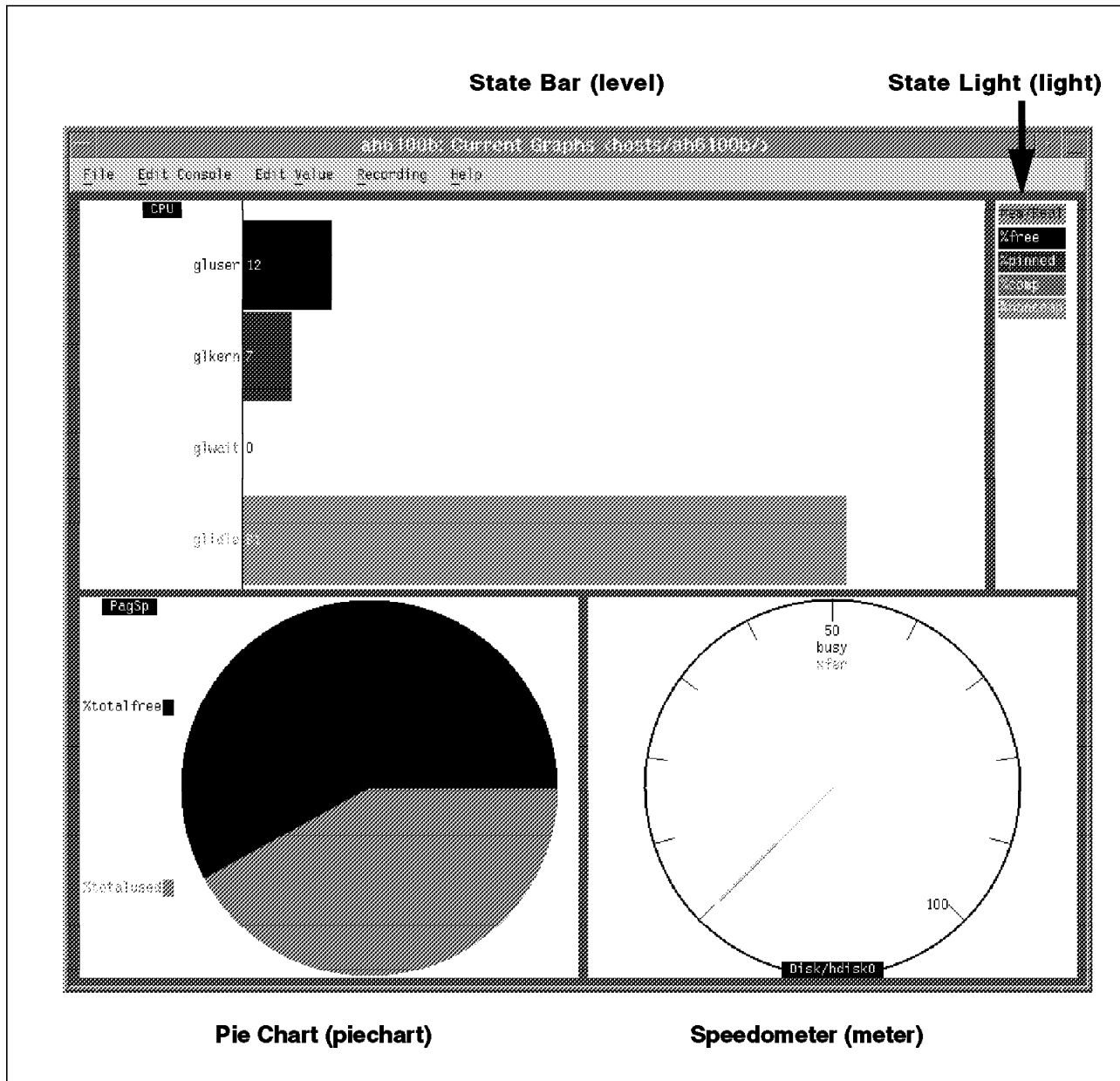


Figure 23. Current Graphs

2.3.7 Menu Bar Commands

In configuring the menu bar, we have three main areas of interest:

1. analy - Analysis menu bar
2. ctrl - Controls menu bar
3. util - Utilities menu bar

Flags are commonly used with these operations. Below is a listing of the flags and what they mean as well as some applications of them in the code.

menu.action.\$parameter: flag%(field 1)% (field 2)% (field 3)%comment

- menu - In which menu bar would you like this located?
- action - A unique name for the action.

- \$parameter - A unique name for the parameter.
- flag - Optional field used to specify the flag to use for the command.
- field 1
 - o - Optional data input
 - r - Required data input
- field 2
 - c - Character input
 - n - Numerical input
 - f - Flag definition line
 - e - A flag extension line
- field 3
 - c - Used to display the initial value in the field as a character
 - n - Used to display the initial value in the field as a number
 - f - Flag extension
 - e - A flag extension line
- comment - Required field. Specifies the prompt text associated with the defined option.

2.3.7.1 Utilities - Menu Bar Pull-down

To customize the utilities menu bar pull-down, the user must first define what he/she would like to see and insert the relevant code into the xmperf.cf configuration file.

Who is Logged on - Example: Below is an easy example of inserting the who command into the utilities menu bar pull-down:

The command runs a basic who -u and prints it to the screen

```
util.Who is logged on.program: aixterm -sb -fg black -bg white -geometry
80x20 -T Who -e ksh -c "(who -$opt ;read)"
util.Who is logged on.$opt: %r%f%% Who Options
util.Who is logged on.$opt: u%O%e%y% Activity and PID of shell
util.Who is logged on.$opt: q%O%e%% Quick (only user and host name)
util.Who is logged on.$opt: a%O%e%% All (AbdHlprTtu) options
util.Who is logged on.$opt: l%O%e%% Login processes
```

Note

Always comment the original and make a copy in your xmperf.cf configuration file to help you with any difficulties later on.

Below is a breakdown of the code that was inserted into xmperf.cf:

util Defines the utilization menu pull-down.

Who is logged on Defines the title of the action.

program: Defines that it is a program to be run.

aixterm -sb -fg black -bg white -geometry

80x20 -T Who -e ksh -c "(who -\$opt ;read)"

- aixterm : Defines an aixterm to start up.
- -sb : Scroll bar set on.

- -fg black : Foreground color black.
- -bg white : Background color white.
- -geometry 80x20 : The size of the aixterm window is 80x20.
- -T Who : Defines a title for the screen created.
- -e ksh : Executes a korn shell command.
- -c "(who -\$opt ;read)" : Runs the command who with the -\$opt option and then waits for the user to press Enter or close the aixterm.

%r%f%% Who Options Defines the title for the actions below.

- r : This field is required.
- f : Defines the field as a flag definition.
- Who Options : Specifies the title to use for the panel.

u%o%e%y% Activity and PID of shell (-u) Defines the flag "u" to use.

- u : Defines the flag "u" to use in the program.
- o : This flag is optional.
- e : A flag extension line.
- y : Selects the first option as default.
- Activity and PID of shell : Summary of the option.

The three options below follow the same context of the above.

q%o%e%% Only user and host name (-q)

a%o%e%% All (AbdHlprTtu) options (-a)

l%o%e%% Login processes (-l)

When you restart xmperv you will see that in the Utilities pull-down you can now see the option Who is logged on:

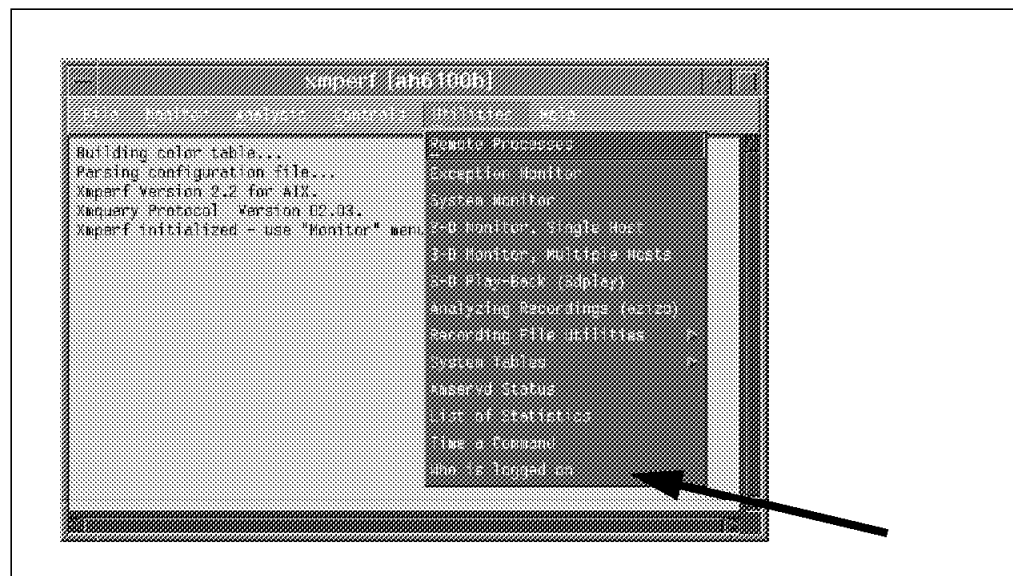


Figure 24. Who is Logged on Pull-down

After you select **Who is logged on**, the following appears:



Figure 25. The Who is Logged on Panel

To see who is logged on to the system, let's select the **Proceed** button. The following is output to your screen depending on who is logged on at that time:

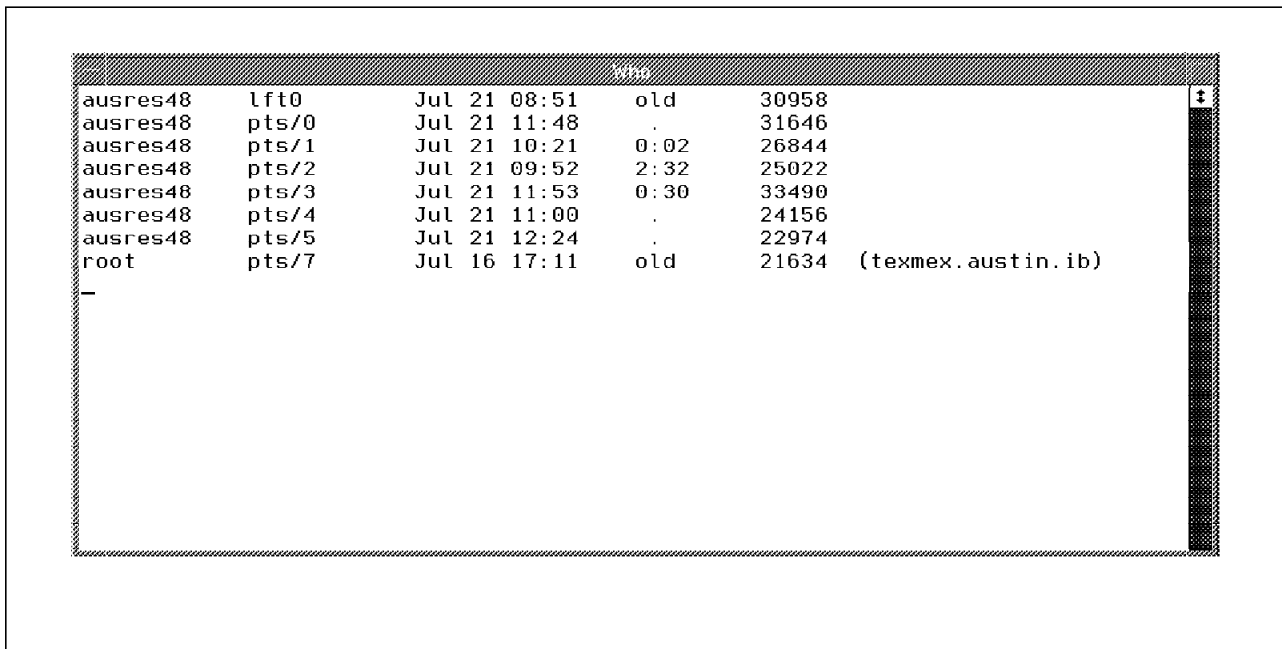


Figure 26. The Who is Logged on Output Screen

Basic df - Example: The xmperv.cf configuration file was then tailored for the df command by using the following segment of code:

The command runs a basic df and allows options then prints it to the screen

```
util.Check FileSystem Sizes.program: aixterm -sb -fg black -bg white -
geometry 80x20 -T Disk -e ksh -c "(df -$opt $filesystem;read)"
util.Check FileSystem Sizes.$opt:      %r%f%%Options
util.Check FileSystem Sizes.$opt:      k%o%e%y% 1024 Blocks
util.Check FileSystem Sizes.$opt:      P%o%e%y% 512 Blocks
util.Check FileSystem Sizes.$filesystem: %o%c%% FileSystem
```

When you restart xmperv, you will see that in the Utilities pull-down you can now see the option Check FileSystem Sizes:

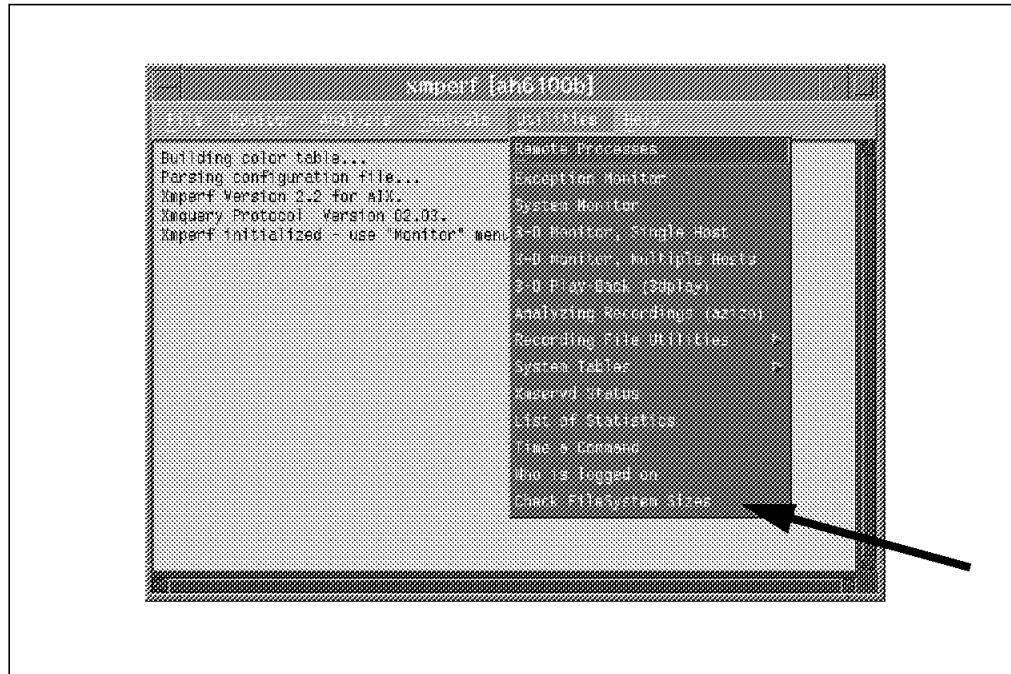


Figure 27. Check FileSystem Sizes Pull-down

The output after **Check FileSystem Sizes** is selected is as follows:

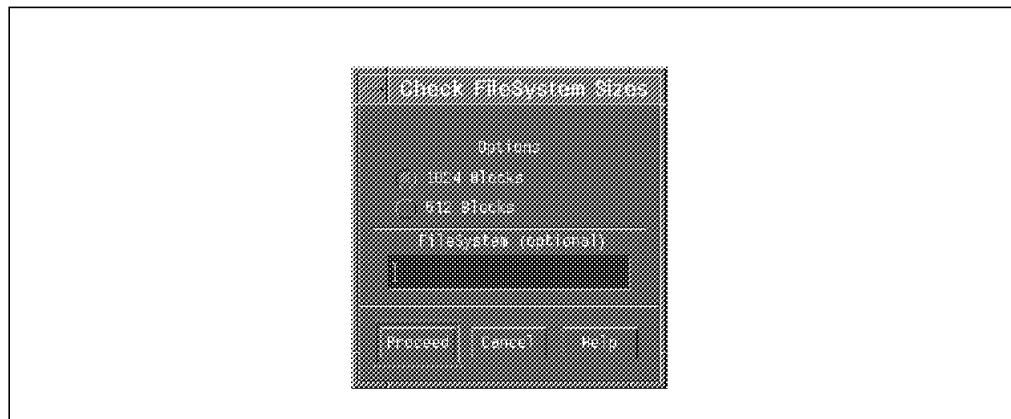


Figure 28. The Check Filesystem Sizes Panel

After you select **Proceed**, the following is output to your screen:

| Filesystem | 1024-blocks | Free | %Used | Iused | %Iused | Mounted on |
|-------------|-------------|----------|-------|-------|--------|------------------------|
| /dev/hd4 | 12288 | 2416 | 81% | 1652 | 21% | / |
| /dev/hd2 | 626688 | 19848 | 97% | 18795 | 12% | /usr |
| /dev/hd9var | 12288 | 10652 | 14% | 233 | 6% | /var |
| /dev/hd3 | 12288 | 8860 | 28% | 98 | 3% | /tmp |
| /dev/hd1 | 28672 | 4820 | 84% | 1470 | 18% | /home |
| /dev/lv00 | 24576 | 8648 | 65% | 1619 | 27% | /usr/vice/cache |
| /dev/lv03 | 151552 | 20580 | 87% | 1340 | 4% | /opt/lotus |
| /dev/lv01 | 12288 | 10548 | 15% | 83 | 3% | /var/dce |
| /dev/lv02 | 40960 | 8988 | 79% | 2019 | 20% | /var/dce/adm/dfs/cache |
| DFS | 9000000 | 9000000 | 0% | 0 | 0% | /... |
| AFS | 72000000 | 72000000 | 0% | 0 | 0% | /afs |
| /dev/lv04 | 16384 | 7848 | 53% | 80 | 2% | /home/ausres48/example |
| /dev/lv05 | 45056 | 43604 | 4% | 17 | 1% | /testfs |

Figure 29. The Check FileSystem Size Output Screen

2.3.7.2 Analysis - Menu Bar Pull-down

In customizing the analysis menu bar, we use the `iostat` command and the information given to us. We want to change the output so that we can use all options associated with the command. The following was inserted into the `xmperf.cf` configuration file instead of the default `analy.iostat.program` entry.

analy.menubegin.iostat

```
analy.iostat.program: aixterm -n iostat -T iostat -sl 1000 -sb \
    -e ksh -c "(iostat $int $iter; read)" &
analy.iostat.$int:    %r%n%5%Interval between samples
analy.iostat.$iter:  %o%n%20%Number of samples
```

The command runs a iostat command with all options

```
analy.iostat options.program: aixterm -n iostat -T iostat -fg black \
    -bg white -sl 1000 -sb -e ksh -c "(iostat -$opt $disk $int $iter; read)" &
analy.iostat options.$opt:    %r%f%%Report Type
analy.iostat options.$opt:    d%o%e%y% drive report
analy.iostat options.$opt:    t%o%e%y% tty/cpu report
analy.iostat options.$disk:  %o%c%% Disk to monitor (default all)
analy.iostat options.$int:  %o%n%1% Interval between samples
analy.iostat options.$iter:  %o%n%10% Number of samples
analy.menuend.analy
```

Below is a breakdown of the individual parameters newly inserted into the configuration file:

analy.menubegin.iostat

- `analy` : Defines the analysis menu pull-down
- `menubegin` : Start a new menu within the pull-down
- `iostat` : Calls this menu option `iostat`

analy.iostat options.program:

- `iostat options` - Defines the title of the action
- `program` - Defines that it is a program to be run

`aixterm -n iostat -T iostat -fg black`

`-bg white -sl 1000 -sb -e ksh -c "(iostat -$opt $disk $int $iter; read)"`

- `aixterm` : Defines an aixterm to startup.
- `-n iostat` : Defines the name.
- `-T iostat` : Defines the title to use.
- `-fg black` : Foreground color is black.
- `-bg white` : Background color is white.
- `:` : Defines a new line.
- `-sl 1000` : Save 1000 lines.
- `-sb` : Scroll bar is set on.
- `-e ksh -c` : Executes a korn shell command
- `iostat` : Executes the command `iostat`.
- `$opt` : Select the `-d` (drive report) or `-t` (tty/cpu report) options for `iostat`.
- `$disk` : Select the disk to monitor (default is all disks).
- `$int` : Interval between samples (default value given = 1).
- `$iter` : Number of samples (default value given = 10).
- `read` : Waits for the user to hit enter or close the screen.

`analy.iostat options.$opt: %r%f%%Report Type`

- `$opt` : Flag extension used.
- `r` : Required field.
- `f` : Flag extension.
- `Report Type` : Title.

The following five options follow the same pattern as described above:

`analy.iostat options.$opt: d%%e%%y% drive report`

`analy.iostat options.$opt: t%%e%% tty/cpu report`

`analy.iostat options.$disk: %%c%% Disk to monitor (default all)`

`analy.iostat options.$int: %%n%1% Interval between samples`

`analy.iostat options.$iter: %%n%10% Number of samples`

The last line closes down the menu bar pull-down:

`analy.menuend.analy`

If you now start `xmperf` and go into the Analysis pull-down, in the IO Analysis section you will see what we just defined, namely the `iostat` (1). Within this selection, you will see the default definition of the `iostat` (2) command and the newly inserted `iostat options` (3) command:

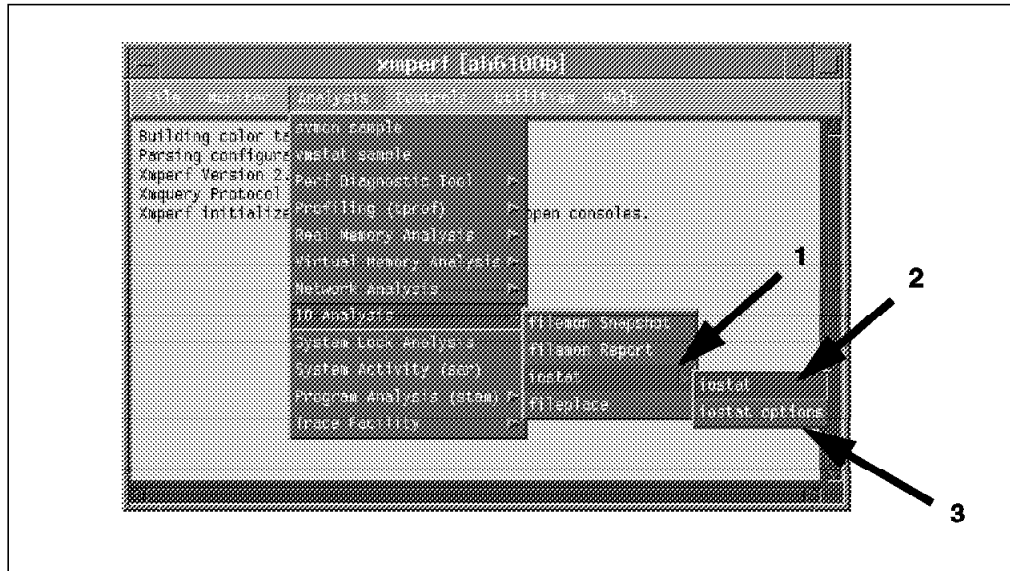


Figure 30. The iostat Options Pull-down

After selecting **iostat options**, the following display will appear. You may now type in the relevant data that you would like to display or leave it blank depending on your configuration required. We wish to monitor the hdisk0 disk usage by inserting hdisk0 in the Disk to monitor (default: all) (optional) field and leaving the other fields as the defaults.

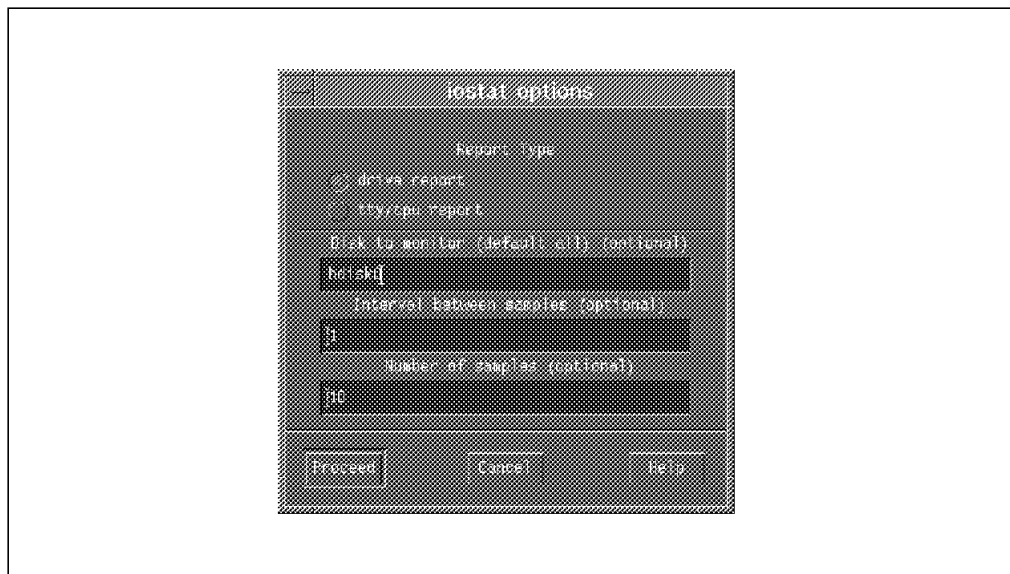
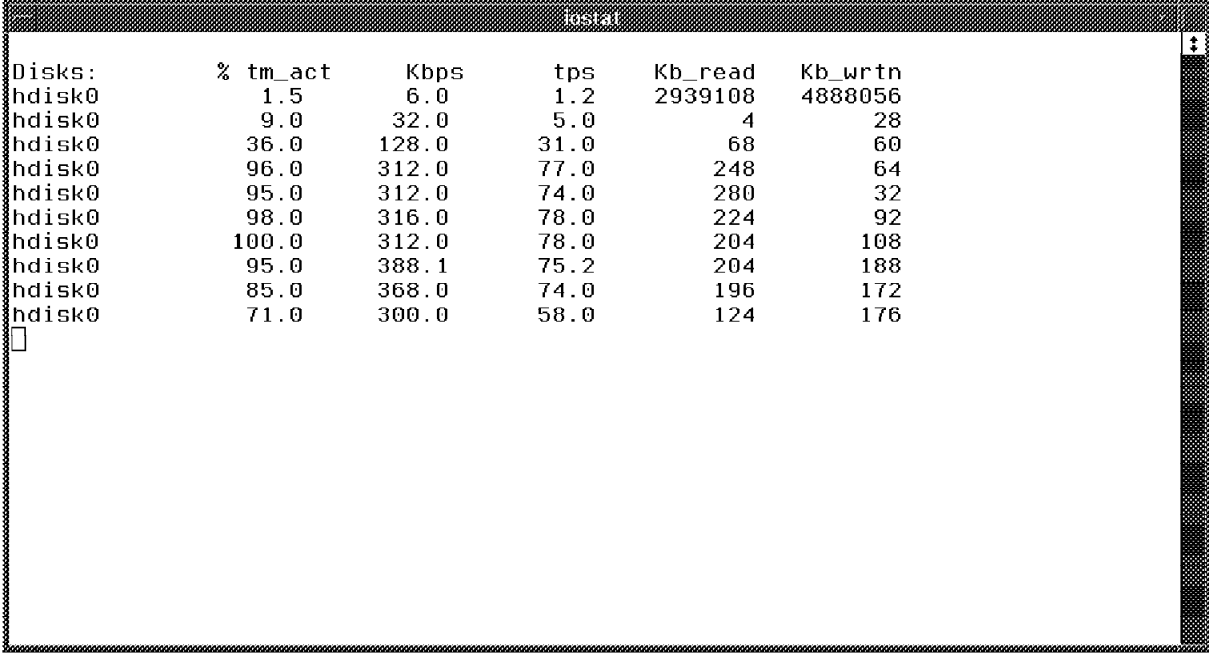


Figure 31. The iostat Options Panel - Complete

Now when you click on **Proceed**, the following output appears on your screen:



```
iostat
Disks:      % tm_act   Kbps      tps     Kb_read  Kb_wrtn
hdisk0      1.5         6.0       1.2    2939108  4888056
hdisk0      9.0        32.0       5.0         4         28
hdisk0     36.0       128.0     31.0         68         60
hdisk0     96.0       312.0     77.0        248         64
hdisk0     95.0       312.0     74.0        280         32
hdisk0     98.0       316.0     78.0        224         92
hdisk0    100.0       312.0     78.0        204        108
hdisk0     95.0       388.1     75.2        204        188
hdisk0     85.0       368.0     74.0        196        172
hdisk0     71.0       300.0     58.0        124        176
```

Figure 32. The iostat Options Output

2.3.7.3 Controls - Menu Bar Pull-down

In customizing the controls menu bar, we will look into the option of increasing the file system size. The following was inserted into the xmpervf.cf configuration file with a brief breakdown of the action:

Comment on the modifications made:

```
# This command increases the size of the filesystem specified.
```

Initialize a menu bar pull-down within the Control pull-down called FileSystem Controls:

```
ctrl.menubegin.FileSystem Controls
```

Part to Check for File System Usage: This section initializes the aixterm to start up, run the df -k command and then wait for an Enter key or close of the window:

```
ctrl.FileSystem Summary.program:    aixterm -n Size -T Size -fg black
-bg white -sl 1000 -sb -e ksh -c "(df -k; read)" &
```

Part to Increase File System Size: This section initializes an aixterm, but before it runs the chfs command, it prompts the user to logon as root and provide some parameters:

```
ctrl.Increase FileSystem Size.program:    aixterm -geometry 80x10 -n Size
-T Size -fg black -bg white -sl 1000 -sb \
-e ksh -c "(su root '-c (chfs -a size=+$size /$file)'; read)" &
```

This is the first parameter, and it asks the user what factor the filesystem should be increased by (512 KB):

```
ctrl.Increase FileSystem Size.$size:  %r%n%Number of 512 byte blocks to increase by
```

This section prompts the user for which file system to expand:

```
ctrl.Increase FileSystem Size.$file:  %r%c%The name of the file system to increase
```

This ends the menu bar:

```
ctrl.menuend.ctrl
```

After the code is inserted into the xmperv.cf configuration file, you see the following selections in the **FileSystem Controls (1)** pull-down: **FileSystem Summary (2)** and **Increase FileSystem Size (3)**.

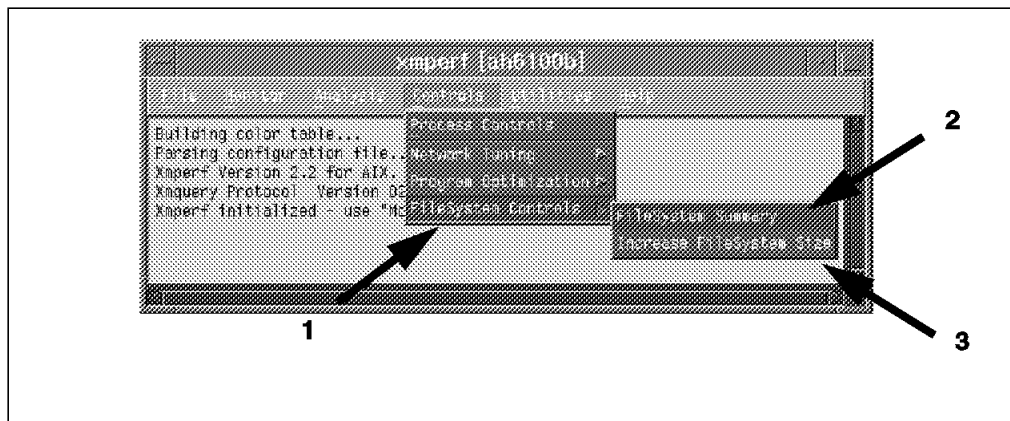


Figure 33. File System Controls Pull-down

When you select **Increase FileSystem Size (3)**, you get the prompt asking by what size (512 KB) to increase the file system and which file system to increase. In this particular example, we increase the File System **testfs** by a factor of **10** x 512 KB blocks:

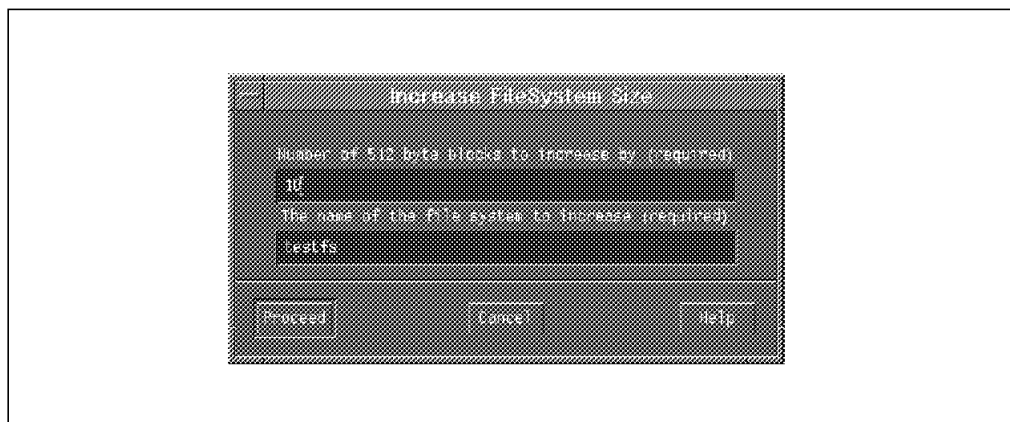


Figure 34. File System Controls - Increase File System Size Inputs

After clicking on the **Proceed** button, you are prompted for root's password if you are not logged in as root already. The following then appears on your screen:

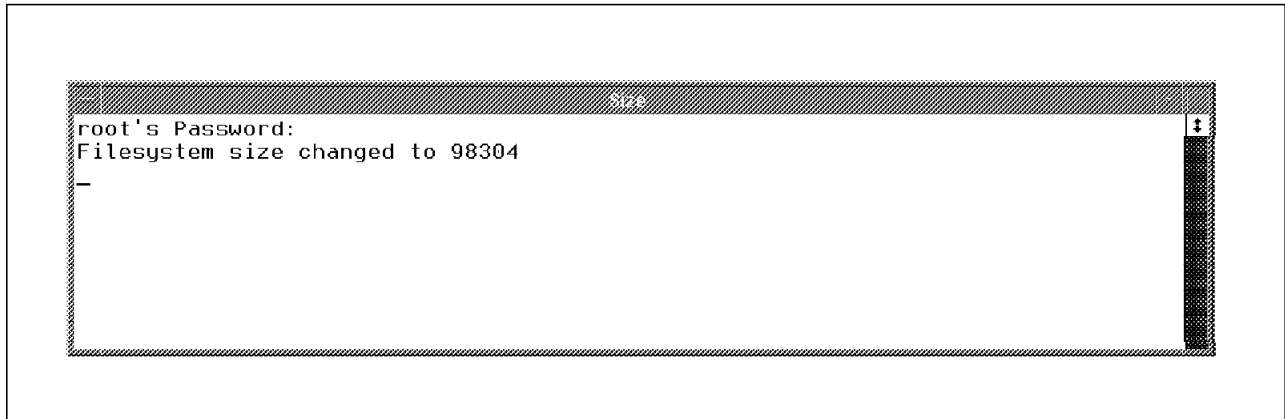


Figure 35. File System Controls - Increase File System Size Output

2.3.7.4 Process Controls

The process controls section in xmpert is used to administer individual processes that are running. Process Controls is accessed from the Controls pull-down in the main xmpert window:

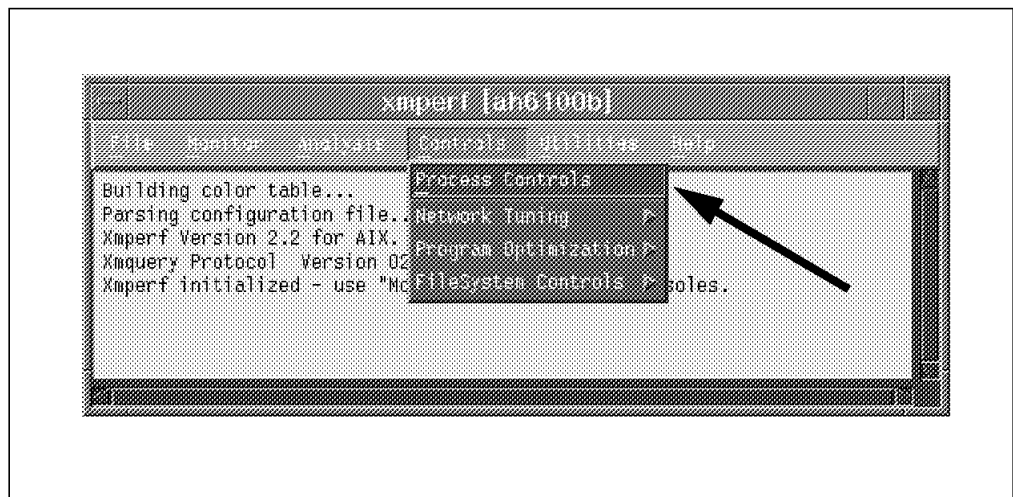


Figure 36. Controls Pull-down

After selecting **Process Controls**, a screen appears with all the active processes and information on them. You may select one or more of them and then click on **Execute**, which will bring up the screen seen in the figure below:

| Controlling Processes on {ah6100b} | | | | | | | | | | | | | | |
|------------------------------------|-------------|-------------------|--------|------|----------|----------------|--------|---------------|-------|---------------|--------|--------------|-------|-------|
| File | Sort | Execute | Down | Help | | | | | | | | | | |
| Process ID (PID) | --Command | svmon Report | Effect | ID | User-ID | --4 KB Pages-- | Latest | -CPU Seconds- | Accum | -Page-Faults- | Parent | Relationship | | |
| | | Increase Priority | | | | Data | Text | Page | CPU | Process | Total | I/O | Other | |
| | | Decrease Priority | | | | Res | Res | Spac | Perct | Total | Total | | | |
| 30324 | xmperf | Kill Process(es) | | 648 | root | 423 | 61 | 381 | 9.4 | 1 | 1 | 119 | 547 | ##### |
| 22618 | netscape | | | 648 | ausres48 | 275 | 143 | 2798 | 4.8 | 1330 | 1380 | 2584 | 3561 | ### |
| 27510 | xv | | | 64 | ausres48 | 244 | 200 | 156 | 1.9 | 0 | 0 | 2 | 299 | ##### |
| 2274 | X | | | 60 | root | 1568 | 158 | 3270 | 1.1 | 5125 | 5125 | 10369 | 33594 | ### |
| 31002 | maker5X.exe | | | 60 | ausres48 | 2368 | 926 | 3267 | 0.9 | 252 | 252 | 9074 | 8723 | ### |
| 22476 | xmperf | | | 64 | ausres48 | 151 | 61 | 450 | 0.7 | 161 | 161 | 525 | 821 | ##### |
| 1032 | gll | | | 37 | root | 14 | 0 | 14 | 0.6 | 2784 | 2784 | 0 | 13 | ## |
| 26582 | 3dmon | | | 64 | ausres48 | 97 | 8 | 173 | 0.2 | 44 | 44 | 16 | 269 | ##### |
| 25844 | xmservd | | | 24 | root | 38 | 15 | 79 | 0.1 | 687 | 687 | 478 | 454 | ##### |
| 4148 | syncd | | | 60 | root | 9 | 1 | 22 | 0.1 | 782 | 782 | 5293 | 79 | ### |
| 13680 | kbio | | | 60 | root | 6 | 0 | 6 | 0.0 | 0 | 0 | 0 | 4 | ## |
| 26330 | cmA# | | | 54 | root | 8 | 0 | 8 | 0.0 | 132 | 132 | 223 | 4 | ### |
| 26050 | exmon | | | 64 | ausres48 | 98 | 8 | 193 | 0.0 | 17 | 17 | 173 | 351 | ### |
| 1982 | lvmb | | | 60 | root | 5 | 0 | 6 | 0.0 | 0 | 0 | 0 | 2 | ## |
| 22564 | fm_misd | | | 60 | ausres48 | 5 | 0 | 24 | 0.0 | 0 | 35 | 40 | 122 | ### |
| 19570 | ttession | | | 60 | ausres48 | 5 | 0 | 125 | 0.0 | 1 | 1 | 189 | 216 | ### |
| 19208 | afsd | | | 26 | root | 8 | 0 | 70 | 0.0 | 4 | 4 | 265 | 243 | ### |
| 18950 | afsd | | | 26 | root | 8 | 0 | 68 | 0.0 | 4 | 4 | 303 | 272 | ### |
| 18692 | afsd | | | 26 | root | 5 | 0 | 66 | 0.0 | 1 | 1 | 319 | 119 | ### |
| 18434 | afsd | | | 26 | root | 8 | 0 | 64 | 0.0 | 5 | 5 | 388 | 290 | ### |

Figure 37. Process Controls

The defaults in the Execute pull-down are:

- svmon Report
- Increase Priority
- Decrease Priority
- Kill Process(es)

Note

The executables only work if at least one process is selected.

The default code that defines these executables is as follows:

```
proc.svmon Report.program:      aixterm -n svmon -T svmon -sl 500 -sb
                                -e ksh -c "(svmon $1 # ; read)" &
proc.svmon Report.$1:          -P%r%f%svmon report output selection
proc.svmon Report.$1:          n%r%e%Non-systems segments only
proc.svmon Report.$1:          s%r%e%Systems segments only
proc.svmon Report.$1:          a%r%e%All segments
proc.Increase Priority.program: renice -n -$pri -p #
proc.Increase Priority.$pri:    %r%n%1%Priority increase
proc.Decrease Priority.program: renice -n $pri -p #
proc.Decrease Priority.$pri:    %r%n%1%Priority decrease
proc.Kill Process(es).program:  kill -9 #
```

Here is a breakdown of the above code:

- proc : Defines the Executable pull-down in the Process Controls section
- svmon Report : Runs a svmon report for the defined process(es)
- Increase Priority : Increases the process(es) priority
- Decrease Priority : Decreases the process(es) priority
- Kill Process(es) : Kills the select process(es)

Note

The # in all fields of the process control defines the process that is selected.

To modify the Execute pull-down menu, follow the same guidelines described in 2.3.7.1, "Utilities - Menu Bar Pull-down" on page 34, in 2.3.7.2, "Analysis - Menu Bar Pull-down" on page 38, or in 2.3.7.3, "Controls - Menu Bar Pull-down" on page 41.

Note

When defining a new option in the Execute pull-down, you have to include the # because it is the process number. This option is required as input; otherwise the action will not work.

2.4 The xmperf.hlp Help File

In PTX help can be accessed in many different ways. On almost every window that comes up, help is available. Below are a few examples of the more common help options:

- Within the help main menu pull-down, you have the options of seeing help on the Main Window, Help on Help, the Help Index and help On Version:

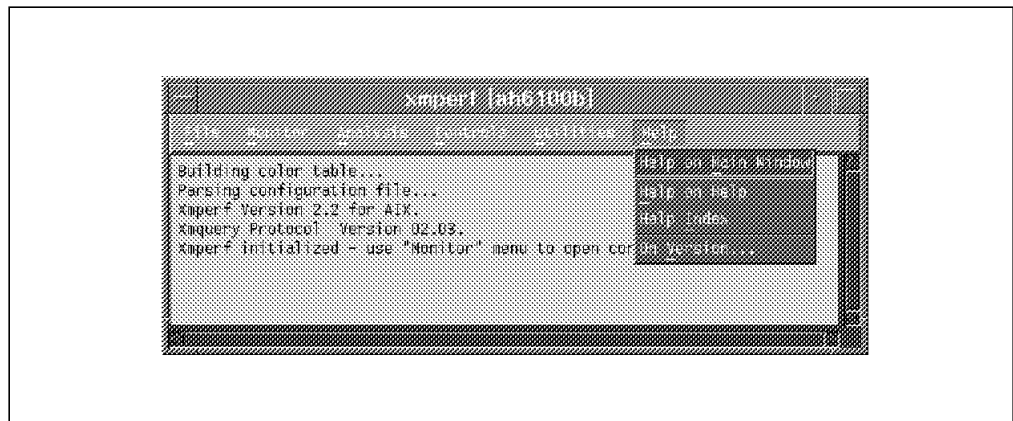


Figure 38. Help Main Menu Pull-down

- When accessing a particular command from the Analysis, Controls, and Utilities pull-downs, you are first prompted with the following screen, where you may then select the help option:

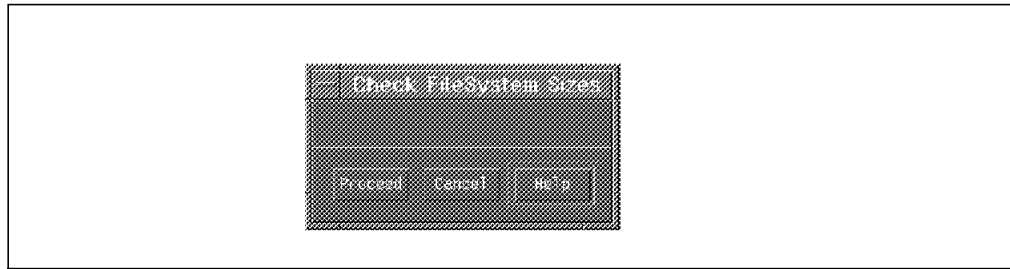


Figure 39. Help Panel

- While using a console, you can select the **Help** option within the Help pull-down.

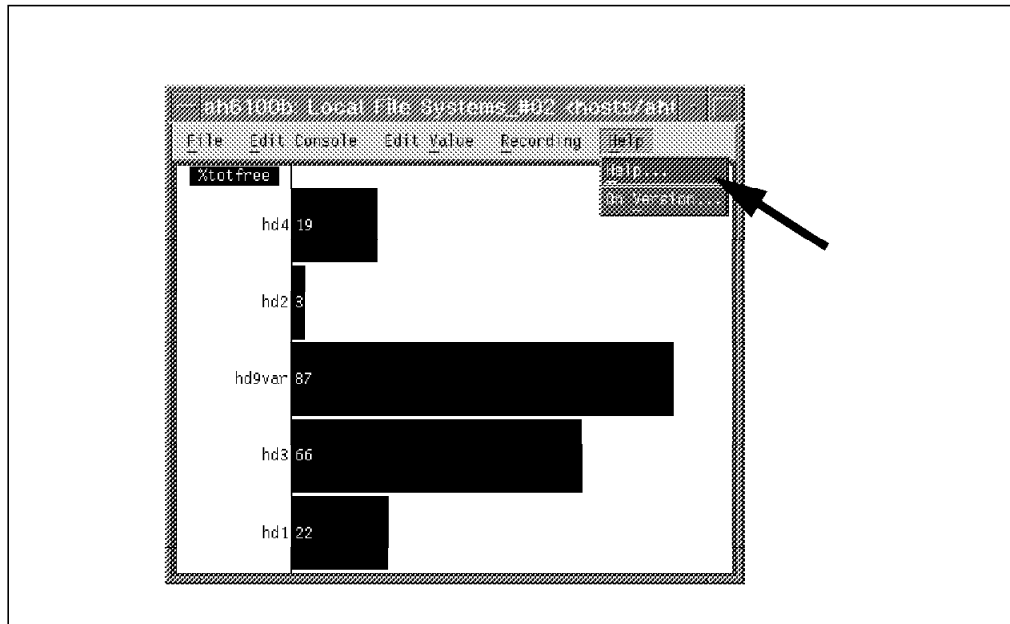


Figure 40. Help Console Pull-down

The help function in xmperv is based upon a simple help file called xmperv.hlp. If this file exists in your home directory, it is used; otherwise it is searched for in your /etc/perf directory. If not found there, it uses the default help file located in /usr/lpp/perfmgr directory.

The following is a segment of code from the default xmperv.hlp file:

```
$help: Main Window
The main window always has a "menubar" at the top. This menubar
defines six pulldown menus, which we shall describe in more detail
in a moment. Before we do so, let's mention the remainder of the
main window. It is used to display messages from xmperv as
necessary.
```

<more>

This segment of code within the xmperv.hlp file then produces the following screen when the user selects **Help on Main Window**:



Figure 41. Help Main Window

The xperf.hlp help file is in the form of:

```
$help: [Action]
[Body]
```

- \$help: - Defines a new help topic within the xperf.hlp file
- Action - The title given to the help panel
- Body - The main body of the help that you would like to see in the panel

The help file contains help texts that are identified by a string of characters. Whenever you select help from a pull-down menu or a dialog window, the help text corresponding to what you are doing is displayed. This identification of help text is done in one of three ways:

1. If you request help for a console, the help text is searched using the console name.

Example:

```
$help: History Graphs
```

History Graphs display the CPU usage on a local system. These graphs consist of all available options of the recording feature in PTX being line, area, skyline and bar. The last example is that of the stacked option which the user may wish to choose, this option adds the current statistics together to total 100%.

After this segment of code was inserted into the xperf.hlp file, the following is output to the screen when the user asks for help:

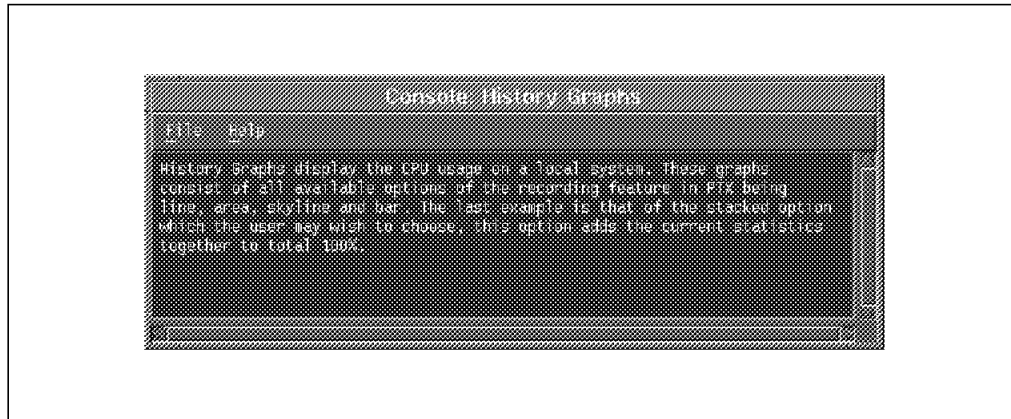


Figure 42. Help On Console - History Graphs

2. If a particular action is a command within xperf, then the user will see the help text displayed when the command panel comes up.

- Example: FileSystem Size as seen in 2.3.7.3, “Controls - Menu Bar Pull-down” on page 41. The following text was inserted into the xperf.hlp file:

```
$help: FileSystem Size
```

```
FileSystem Size uses the df -k command, it shows you the following:
```

```
FileSystem, 1024-blocks,Free,%Used,Iused,%Iused and Mounted on.
```

After inserting this segment of code into the xperf.hlp file and selecting **Help** from the command panel, the following output is produced on the screen:

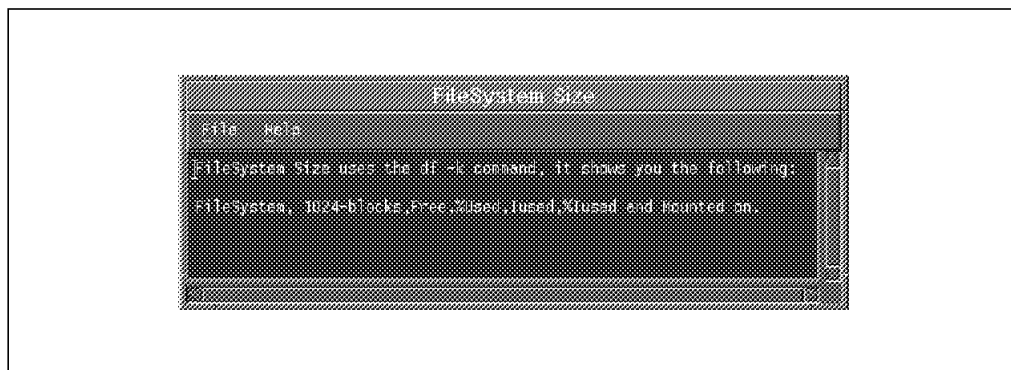


Figure 43. FileSystem Size Help

- Example:

```
$help: iostat options
```

Iostat options allows the user to run the iostat command with various options. These options being iostat -d Disk or iostat -t Disk. This is an extension on what xperf already has with iostat.

note: you will have to know what disk you would like to monitor so if you have any problems with this then use the FileSystem Size command in the utilities pull-down.

After inserting this segment of code into the xmperf.hlp file and selecting **Help** from the command panel, the following output is produced on the screen:

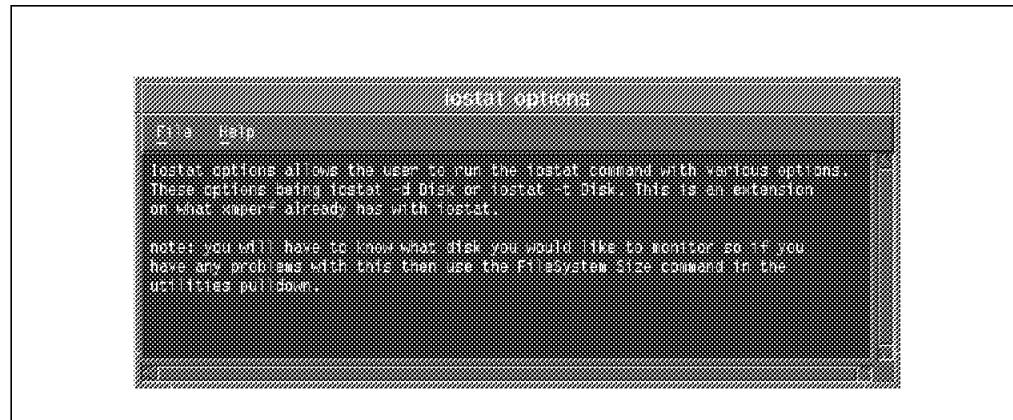


Figure 44. Help On Command - iostat Options

3. In all other cases, the help text to look for is identified by the name of the function you are about to execute.

Example:

\$help: Delete Console

This is a warning dialog box that gives you a last chance to prevent the deletion of a console after you selected the "Delete Console" menu option. It is a safeguard against accidental deletion of consoles.

Click on "OK" if you want to delete the console, otherwise click on "Cancel".

After inserting this segment of code into the xmperf.hlp file and selecting the corresponding **Help** button, the following output is produced on the screen:

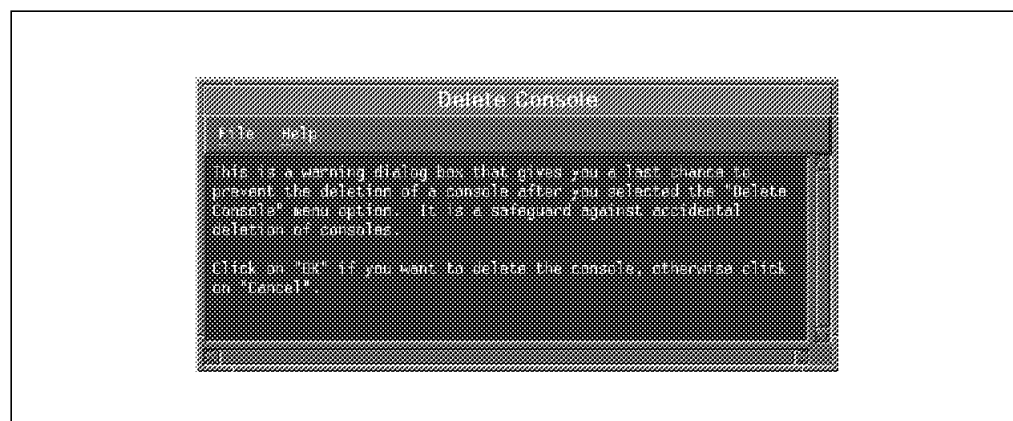


Figure 45. Help On General - Delete Console

2.4.1 Help Index

From any help window, you can get to an index of help text by using the Help pull-down menu and then selecting **Help Index**. The index is a list of all help texts available followed by the action of the text. When you click on a line, a help window is created with the associated help text.



Figure 46. Help Index

Chapter 3. Manager - Advanced Configuration

This chapter goes into the advanced configuration files on PTX manager and shows users how to go in and modify and tailor them for their specific environment or needs. The files are as follows:

- 3Dmon (resource file)
- 3dmon.cf (configuration file)
- 3dplay (command file)
- 3dplay.hlp (help file)
- EXmon (resource file)
- exmon.cf (configuration file)
- exmon.hlp (help file)

When the user goes into PTX and runs a specific command, the tool then goes through the following search path to locate the correct file to use:

1. \$HOME (user's home directory)
2. /etc/perf (unique for all hosts)
3. /usr/lpp/perfmgr (default install directory)

3.1 The 3dmon Configuration

The 3dmon program can be used to monitor up to 576 (24 x 24) statistics simultaneously and display them in the form of a 3D graph.

When you first start 3dmon, you will be prompted to select available hosts.

Note

The hosts that you are able to select depend on the setup of your Rsi.hosts file. For more information, please refer to 2.2, "The Rsi.hosts Host File" on page 13.

For this particular example, there are only four hosts specified in the RSI.hosts file: ah6100a, ah6100b, ah6100c, and itsosmp. Let's select them all:

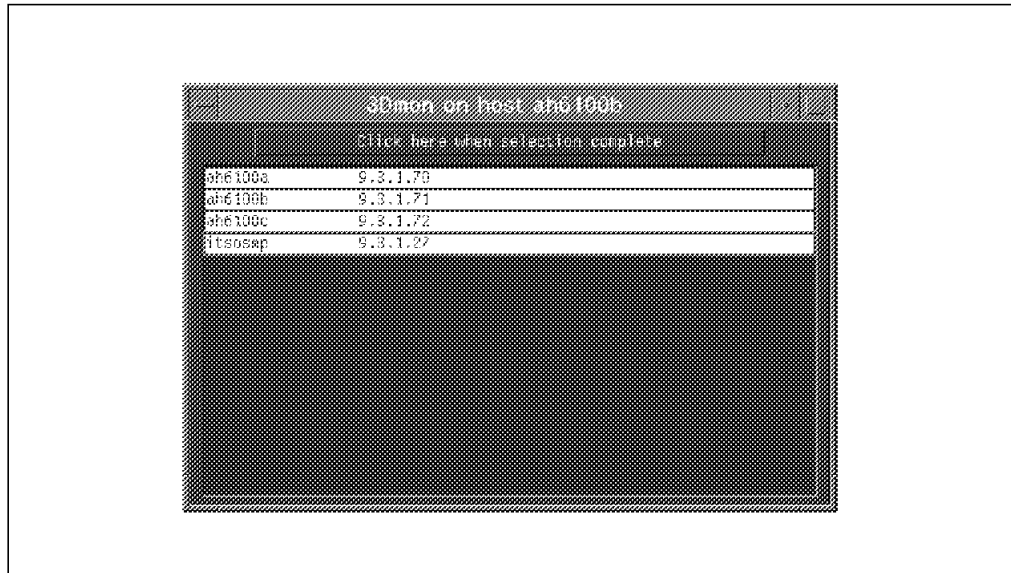


Figure 47. 3dmon - Host Selection Made

After selecting these hosts and pressing **Click here when selection complete**, the following appears on the screen:

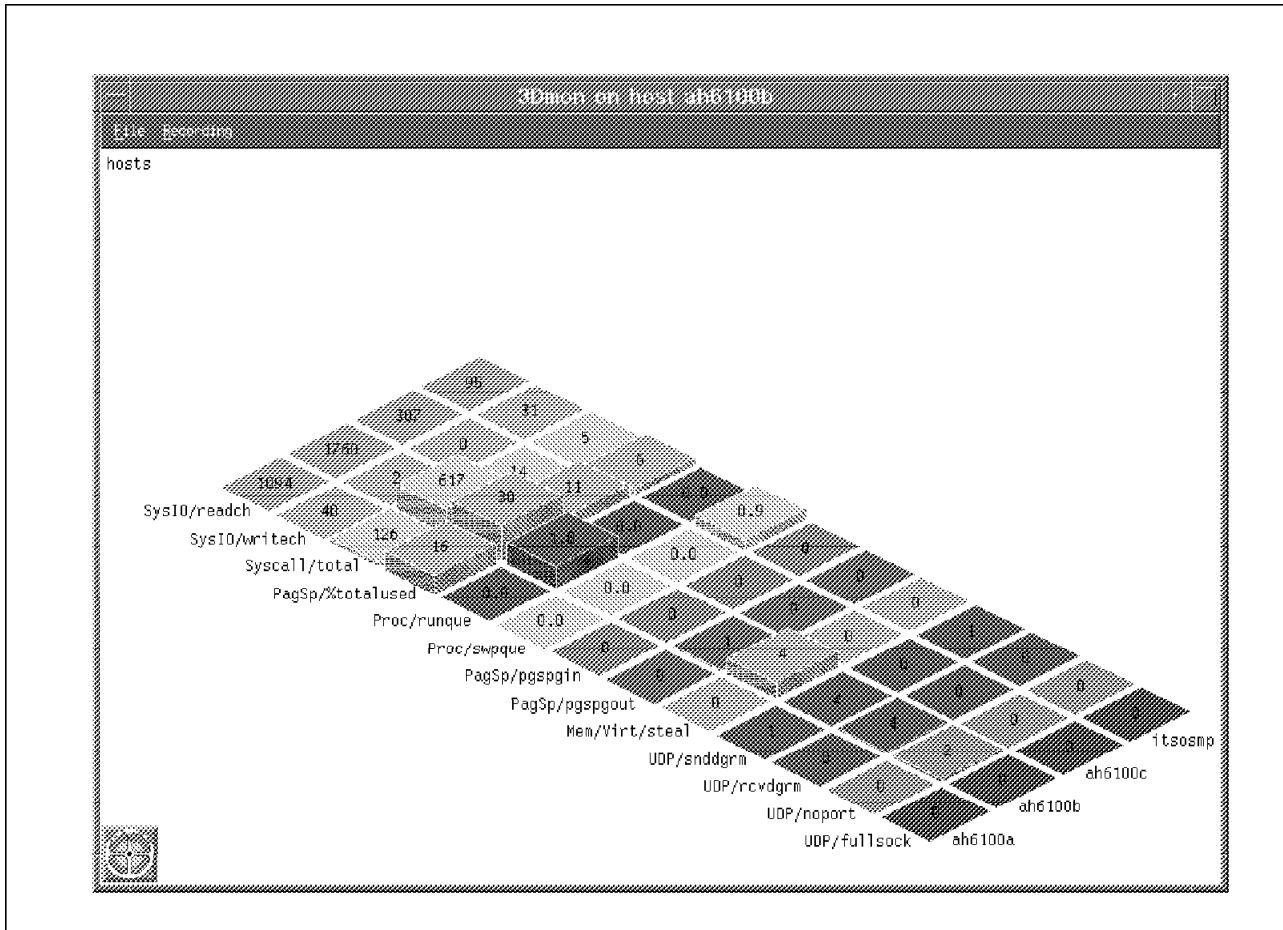


Figure 48. 3dmon - Default Host Configuration

3.1.1 The 3dmon Options

The 3dmon command gives you the opportunity to use different options with the 3dmon start-up. Below are listed some of the more commonly used options in the PTX environment. If you wish to see all the options that you can use with 3dmon, please refer to the *Performance Toolbox for AIX: Guide and Reference*, SC23-2625, Chapter 6.

- v** Verbose. Causes the program to display warning messages about potential errors in the configuration file to stderr. Also causes 3dmon to print a line for each statset created and for each statistic added to the statset, including the results of resynchronizing.
Example: 3dmon -v
- f** File Configuration. Allows you to specify a configuration file name other than the default. If not specified, 3dmon looks for the file in the \$HOME directory. If that file does not exist, the file is searched for in /etc/perf, and last in the /usr/lpp/perfmgr directory.
Example: 3dmon -f 3dmon.config
- i** Sampling interval. If specified, this argument is taken as the number of seconds between sampling of the statistics. If omitted, the sampling interval is 5 seconds. You can specify from 1 to 60 seconds for the sampling interval.
Example: 3dmon -i 1
- h** Host Monitor. Used to specify which host to monitor. This argument is ignored if the specified wildcard is "hosts." If omitted, the local host is assumed. With the PTX Local feature, this flag always uses the local host name.
Example: 3dmon -h itsosmp
- c** Configuration set. When specified, overrides the default configuration set and causes 3dmon to configure its graph using the named configuration set. The argument specified after the -c must match one of the wildcard stanzas in the configuration file. If this argument is omitted, the configuration set used is the first one defined in the configuration file.
Example: 3dmon -c memory
- d** Invitation Delay. Invitation delay. Allows you to control the time 3dmon waits for remote hosts to respond to an invitation. The value must be given in seconds and defaults to 10 seconds. Use this flag if the default value results in the list of hosts being incomplete when you want to monitor remote hosts.
Example: 3dmon -d 30

3.1.2 The 3Dmon Resource File

The 3Dmon resource file is located in the /usr/lib/X11/app-defaults directory and has three options for the user to modify:

1. Font

Default:

```
*GraphFont: -ibm-block-medium-r-normal--15-100-100-100-c-70-iso8859-1
```

Refer to 2.1.1, "The XMperf Font" on page 7, for detailed information on setting a desired font.

2. Background Color

Default:

```
*DrawArea.background:  black
```

If you wish to change the background color in 3dmon, then all you need to do is type over the default color and save the file. Refer to 3.3.1, “The EXmon Resource File” on page 66, for a detailed example of changing the background color.

Example:

```
*DrawArea.background:  white
```

3. Foreground Color

Default:

```
*DrawArea.foreground:  white
```

If you wish to change the foreground color in 3dmon, then all you need to do is type over the default color and save the file. Refer to 3.3.1, “The EXmon Resource File” on page 66, for a detailed example of changing the foreground color. The principle here is the same.

Example:

```
*DrawArea.foreground:  black
```

Note

If you wish to modify this file, then you will need root access to either change the file permissions or modify the file. Always make a backup copy of the original default configuration file.

3.1.3 The 3dmon.cf Configuration File

The 3dmon.cf file is used in conjunction with the 3dmon command and can be tailored to suit your environment needs.

All configuration sets that start with only a local hardware option do not prompt the user for a host to select because it will use the local host. In 3.1.3.1, “The Disk Default Configuration” on page 55, you can see a good example of this.

All other configuration sets that have more than one local selection possible or have more than one wildcard will prompt the user for the host(s) that they would like to monitor. In 3.1.3.2, “The hosts Default Configuration” on page 56, you can see a good example of this.

The 3dmon command may be invoked by using any of the following default configuration sets:

| | |
|------------------|-------------------------------|
| hosts (*) | remote hosts |
| 24 | remote disks, large set |
| Disk | local disks |
| disk (*) | disks on remote hosts |
| Kmem | local kernel memory |
| kmem | kernel memory on remote hosts |
| Proc | local processes |
| proc | processes on remote hosts |
| LAN | local LAN adapters |
| lan | LAN adapters on remote hosts |
| FS | local file systems |

fs (*) file systems on remote hosts
IP/NetiF local IP interfaces
ip IP interfaces on remote hosts
CPU local processors
cpu processors on remote hosts

Note

* indicates the file is explained in the following sections

To invoke 3dmon with a specific configuration set, you need to execute 3dmon with the option `-c`, followed by the configuration set name, `3dmon -c` (configuration set name).

Example: `3dmon -c CPU`

3.1.3.1 The Disk Default Configuration

The *Disk* configuration set is used to monitor local disks on the host. When 3dmon sees that the statistics to be monitored are *busy*, *xfer*, *rblk*, and *wblk*, it recognizes that these are on the local disk; so it will not prompt you to select the desired host(s). The default code for *Disk* is below:

```
wildcard:      Disk                # local disks
busy
xfer
rblk
wblk
```

To execute the *Disk* configuration set, `3dmon -c Disk` was typed in the command window. This will produced the following screen:

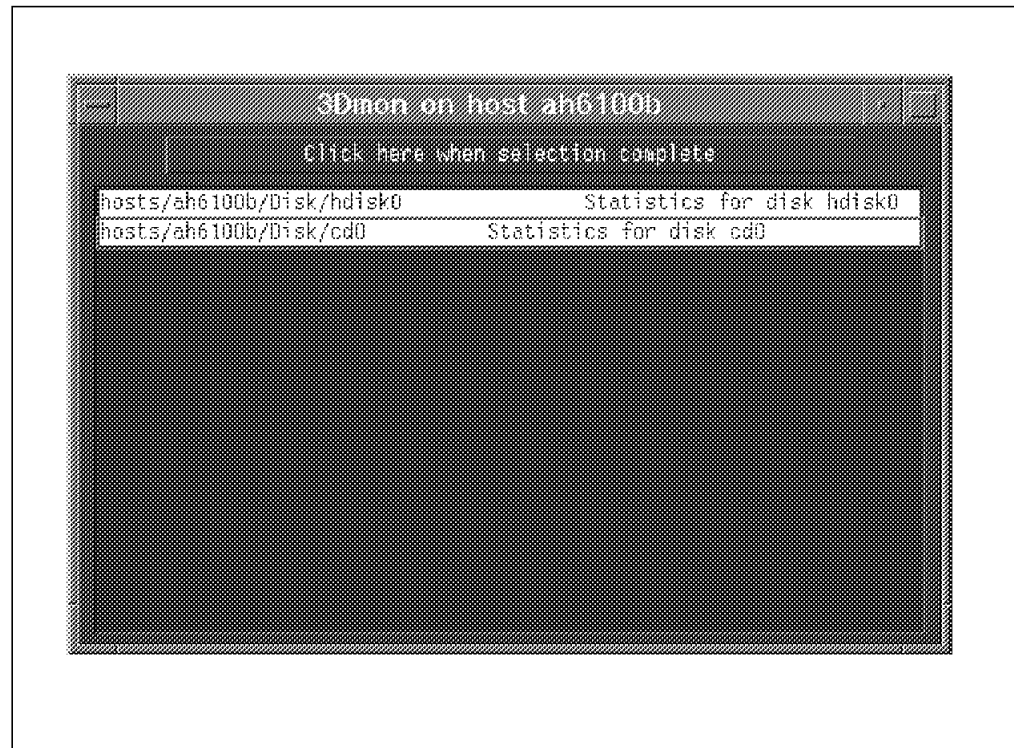


Figure 49. Disk Selection

After selecting both disks to monitor and pressing the **Click here when selection complete**, the following was output to the 3dmon screen:

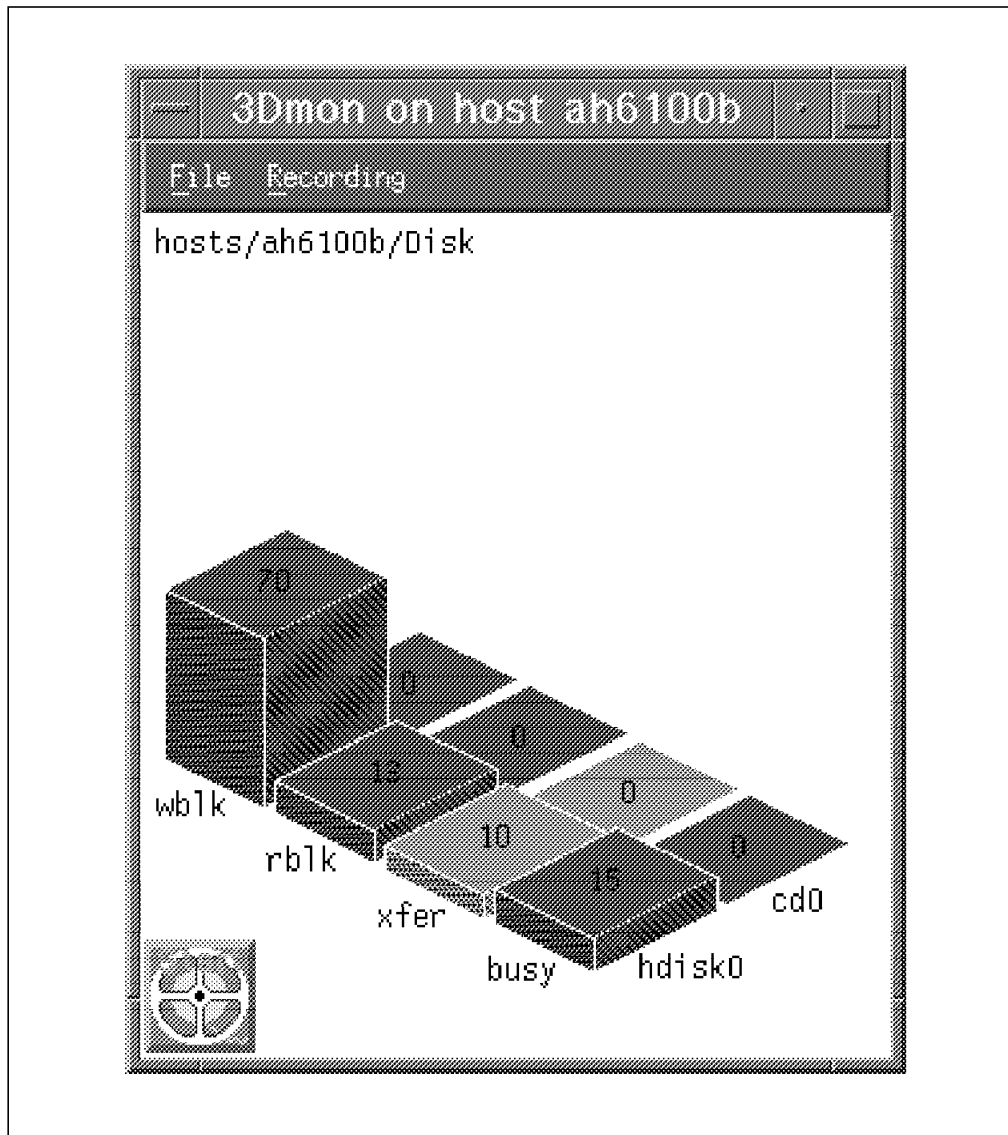


Figure 50. 3dmon - Disk Configuration Output

3.1.3.2 The hosts Default Configuration

If you wish to monitor remote hosts on your network, then you may use the *hosts* configuration. Below is the code from the default 3dmon.cf file:

```
wildcard:      hosts          # remote hosts
UDP/fullsock
UDP/noport
UDP/rcvdgrm
UDP/sniddgrm
Mem/Virt/steal
PagSp/pgspgout
PagSp/pgspgin
Proc/swpque
Proc/runque
PagSp/%totalused
```

```
Syscall/total  
SysIO/writetech  
SysIO/readch
```

Note

Refer to Figure 48 on page 52 to see the output of this file.

3.1.3.3 The disk Default Configuration

The *disk* configuration monitor allows you to monitor remote disks on your network. Below is the code from the default 3dmon.cf file:

```
wildcard:      disk    hosts  # disks on remote hosts  
Disk/*/busy  
Disk/*/xfer  
Disk/*/rb1k  
Disk/*/wb1k
```

Note

Compare the difference between the *disk* and the *Disk* default configuration. In the *Disk* configuration we have a wildcard that allows the user to select more than one host.

After starting up 3dmon with the *disk* configuration (3dmon -c disk), the following screen appears. Let's select the first three hosts:

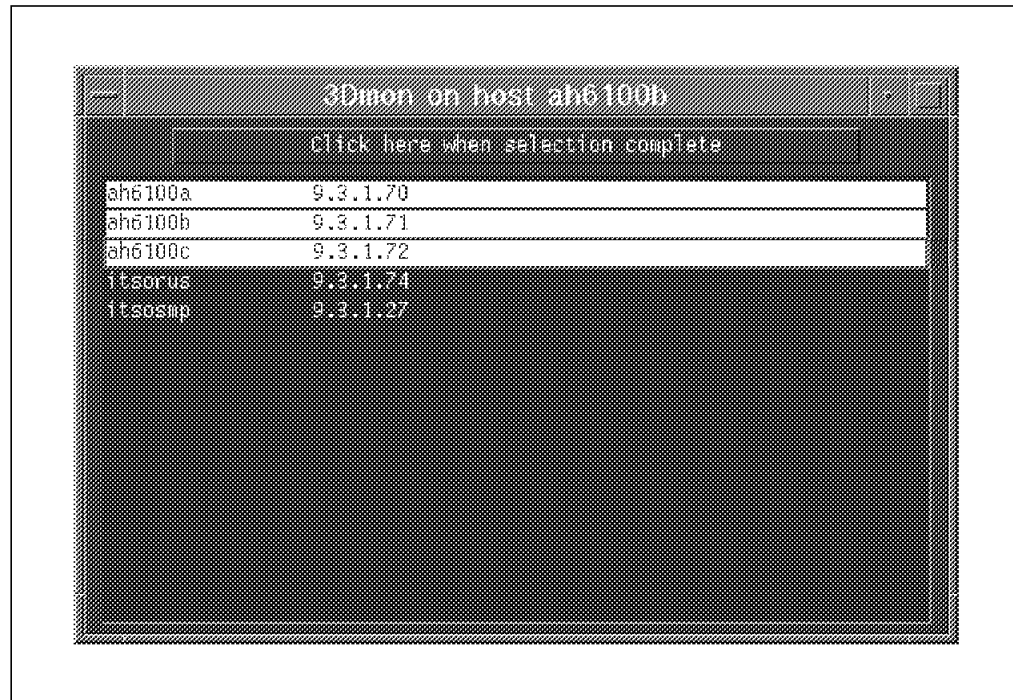


Figure 51. 3dmon - Host disk Selection

The segment of code uses only one wildcard entry:

- Selecting what disk(s) you would like to monitor.

Now let's select **hdisk0** for all three hosts:

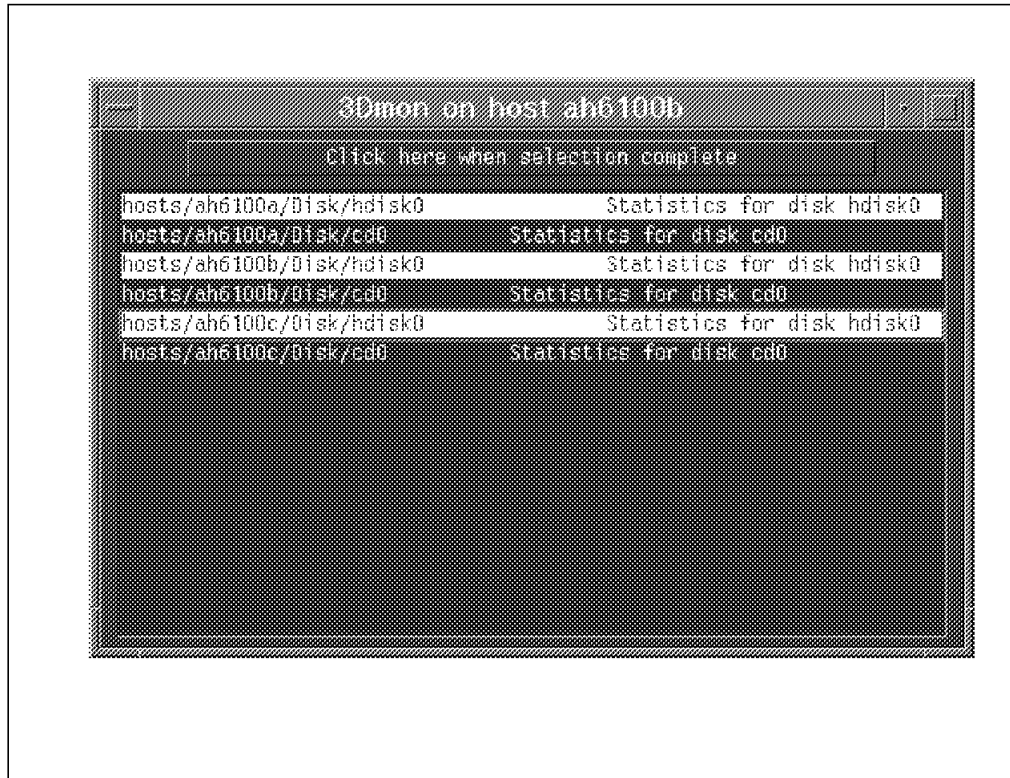


Figure 52. 3dmon - disk Selection

After making the selection and then pressing the **Click here when selection complete**, the following screen appears:

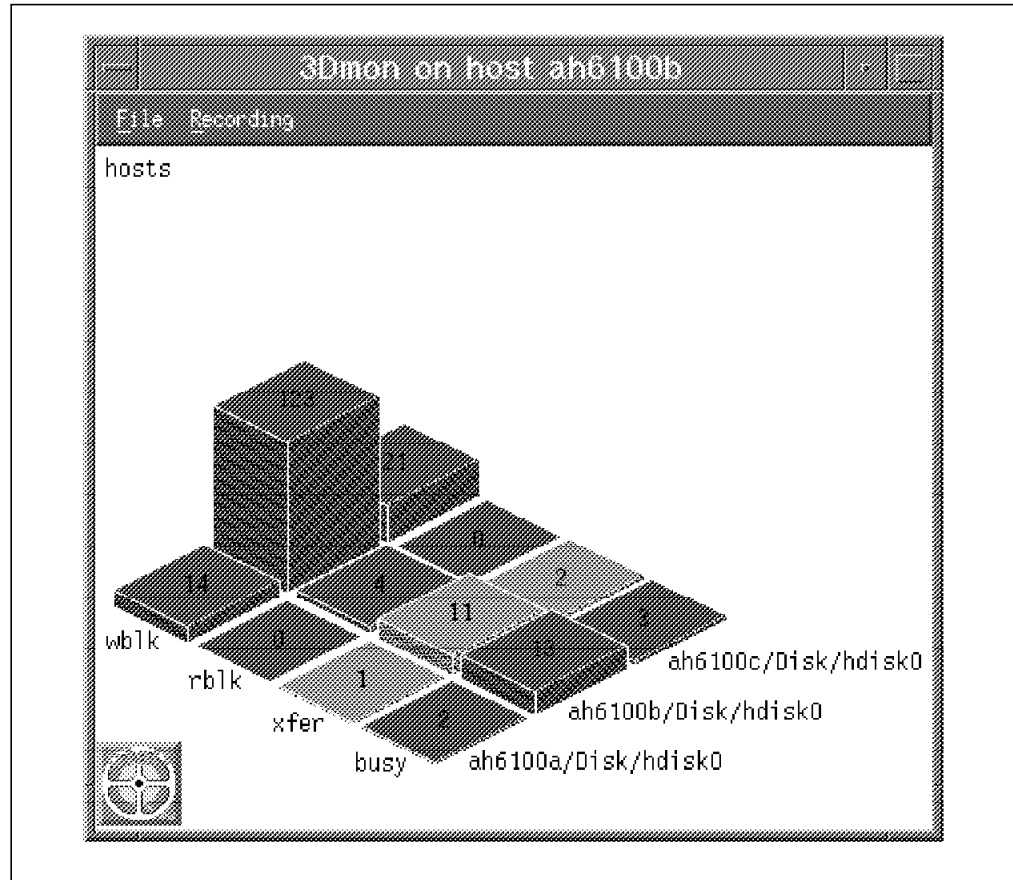


Figure 53. 3dmon - disk

3.1.3.4 The fs Default Configuration

To monitor remote file systems on your network the *fs* configuration is used. Below is the code from the default 3dmon.cf file:

```
wildcard:      fs      hosts # file systems on remote hosts
FS/**/%totfree LV
FS/**/size    LV
FS*/ppsize    VG
FS*/free      VG
FS/iget       FS
```

Note

The FS/**/%totfree indicates a double wildcard selection and is explained below.

After starting up 3dmon with the option (3dmon -c fs), the host selection appears as seen in Figure 51 on page 57. Let's select the first three hosts again.

The *fs* configuration uses two sets of wildcards:

1. The logical volume statistics
2. The volume group statistics

After pressing **Click here when selection complete**, the following appears. Let's select the **/home** and **/home/images** directories from host ah6100b:



Figure 54. 3dmon - fs Selection

Now, pressing **Click here when selection complete** produces the following screen:

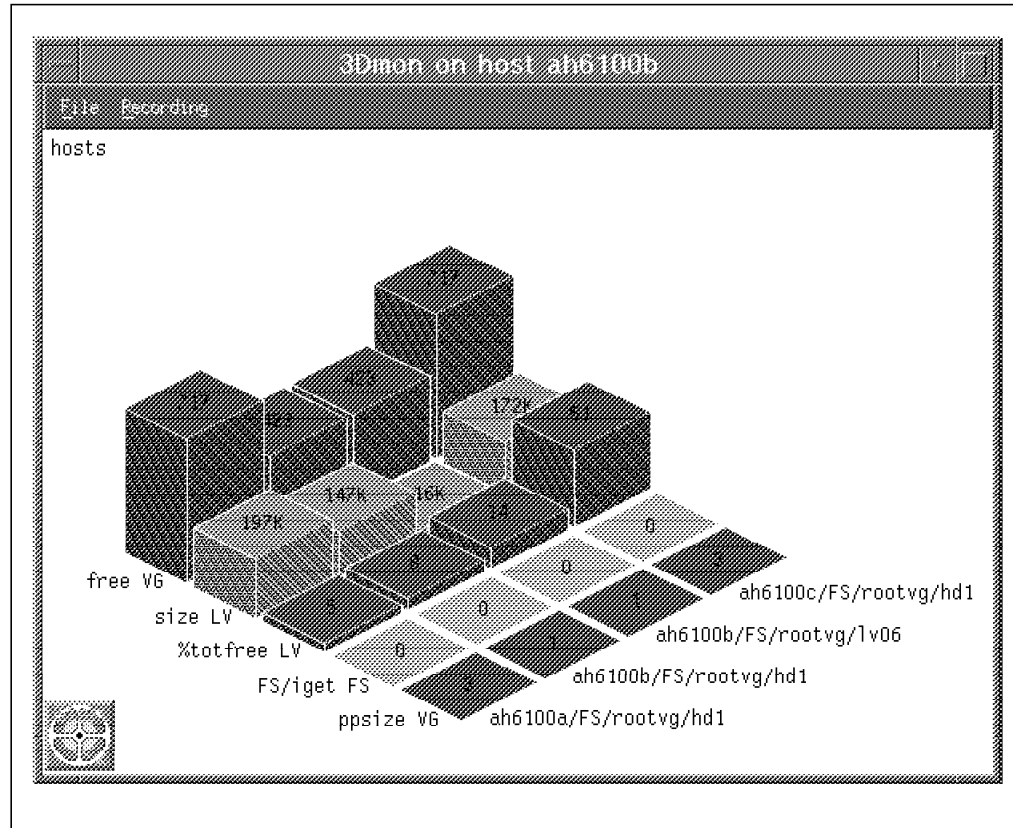


Figure 55. 3dmon - fs Hosts

3.1.4 Customizing the 3dmon.cf File

If you are not happy with the default configuration sets given to you in the 3dmon.cf configuration file, then you may add in extra sets or modify existing ones to your needs. To see a list of available statistics, you can use execute the `xmpeek -l | pg` command, and you will see a list as follows:

| | |
|-----------------------|---|
| /ah6100b/CPU/ | Central processor statistics |
| /ah6100b/CPU/gluser | System-wide time executing in user mode (percent) |
| /ah6100b/CPU/glkern | System-wide time executing in kernel mode (percent) |
| /ah6100b/CPU/glwait | System-wide time waiting for IO (percent) |
| /ah6100b/CPU/glide | System-wide time CPU is idle (percent) |
| /ah6100b/CPU/gluticks | System-wide CPU ticks executing in user mode |
| /ah6100b/CPU/glticks | System-wide CPU ticks executing in kernel mode |
| /ah6100b/CPU/glwticks | System-wide CPU ticks waiting for IO |
| /ah6100b/CPU/gliticks | System-wide CPU ticks while CPU is idle |

Note

This is a very small segment of the output from `xmpeek -l`. The `| pg` option was used so that you may page through all statistics available.

3.1.4.1 Remote CPU Activity

To monitor the main CPU statistics on all hosts, here is the code to place in your 3dmon.cf file:

```
wildcard: HOSTCPU hosts          # Host CPU statistics
CPU/*/user
CPU/*/kern
CPU/*/wait
CPU/*/idle
```

After inserting the code into the 3dmon.cf configuration file and saving it, run the command `3dmon -c HOSTCPU`.

You will now be prompted with a screen asking for the host(s) to monitor as displayed in Figure 51 on page 57. The wildcard in the code allows you to select this. Now let's select the first three hosts again. The ability to select all the desired CPU's is now output to your screen once the desired hosts are selected and **Click here when selection complete** is pressed:

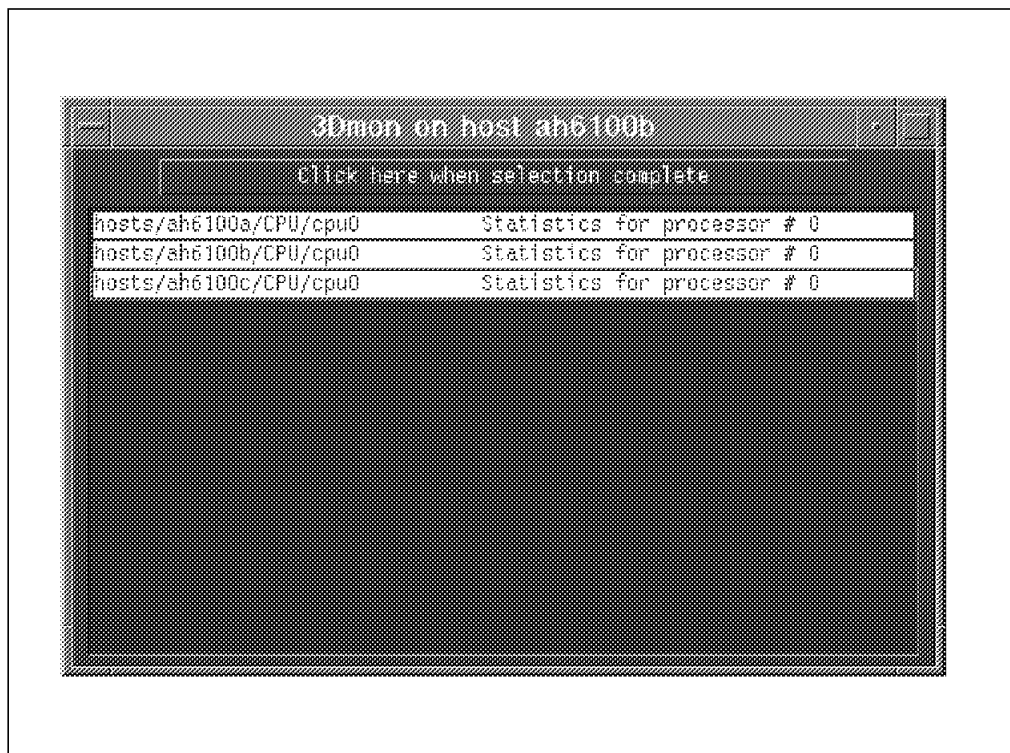


Figure 56. 3dmon - CPU Selection

After selecting all the CPU's and then pressing the **Click here when selection complete** button, the following is output to your screen:

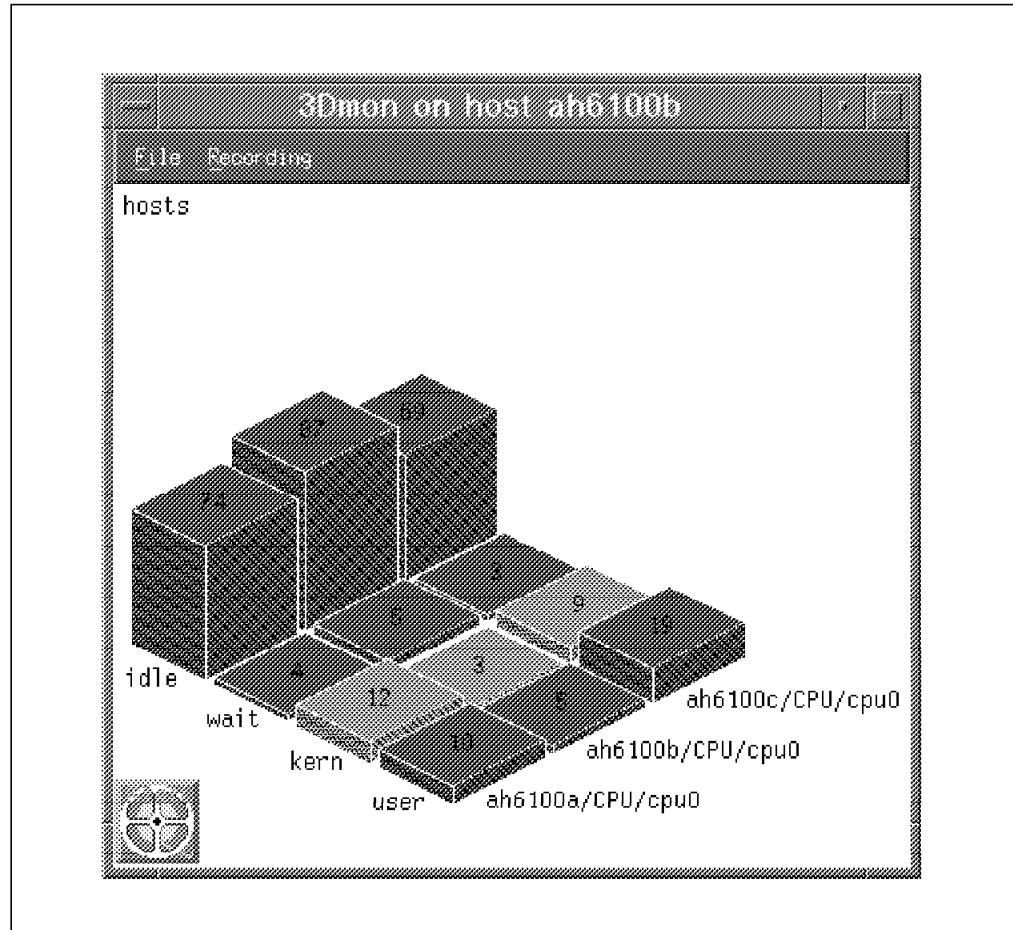


Figure 57. 3dmon - HOSTCPU CPU Output

3.1.4.2 The Multifunction Monitor

To monitor CPU activity along with disk activity and file system activity, this set is used to monitor all these within a specified region. Below is the code used:

Note

You may use the wildcard option with only one of these features. You will get a clash between the settings if you try more than one. For example, you may only select the CPU as a wildcard and not the Disk or FS. Please note that our example has no wildcard specified!

```
wildcard: multi hosts          # remote hosts, large set
CPU/glkern
CPU/gluser
CPU/glwait
CPU/glidle
CPU Activity
Disk/hdisk0/busy
Disk/hdisk0/xfer
Disk/hdisk0/wblk
Disk/hdisk0/rblk
Disk Activity
FS/iget FS
```

FS/rootvg/ppsize VG
FS/rootvg/hd1/%totfree LV
FS/rootvg/hd1/size LV
FS/rootvg/free VG
File Systems

Note

CPU Activity, Disk Activity and File Systems are used as titles for each section, to break them up. The 3dmon command does not recognize these commands and will print a line of blank squares.

After executing the command `3dmon -c multi`, the following is output to your screen. Let's monitor ah6100a, ah6100b, ah6100c, and itsosmp:

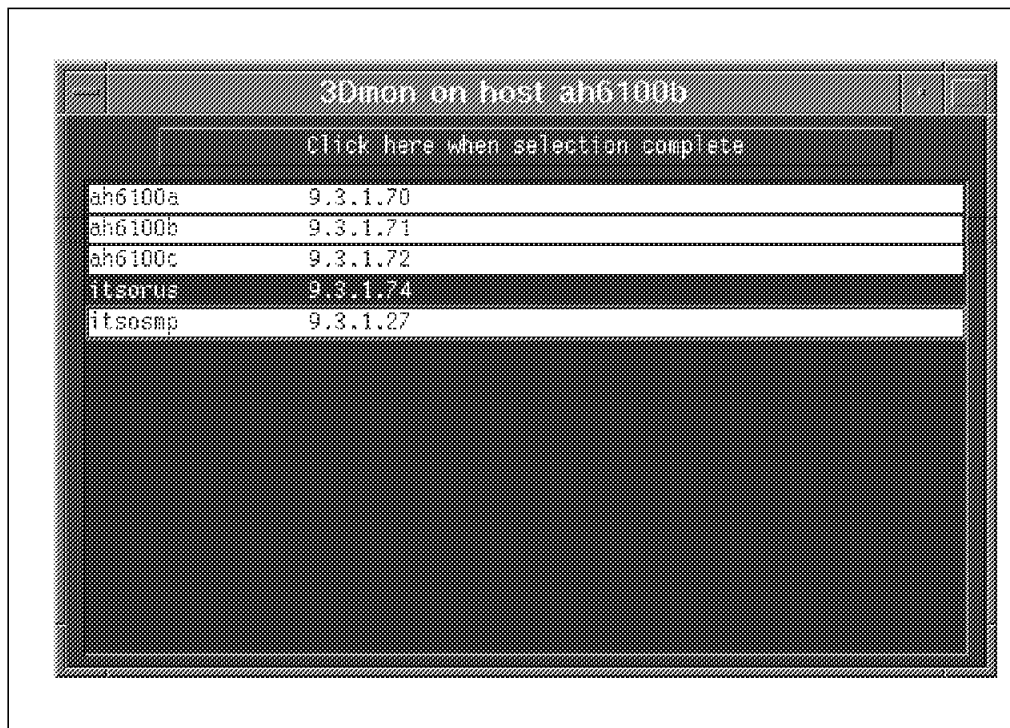


Figure 58. 3dmon - multi host Selection

After selecting these hosts from the wildcard option and pressing the **Click here when selection complete**, the following is displayed on your screen:

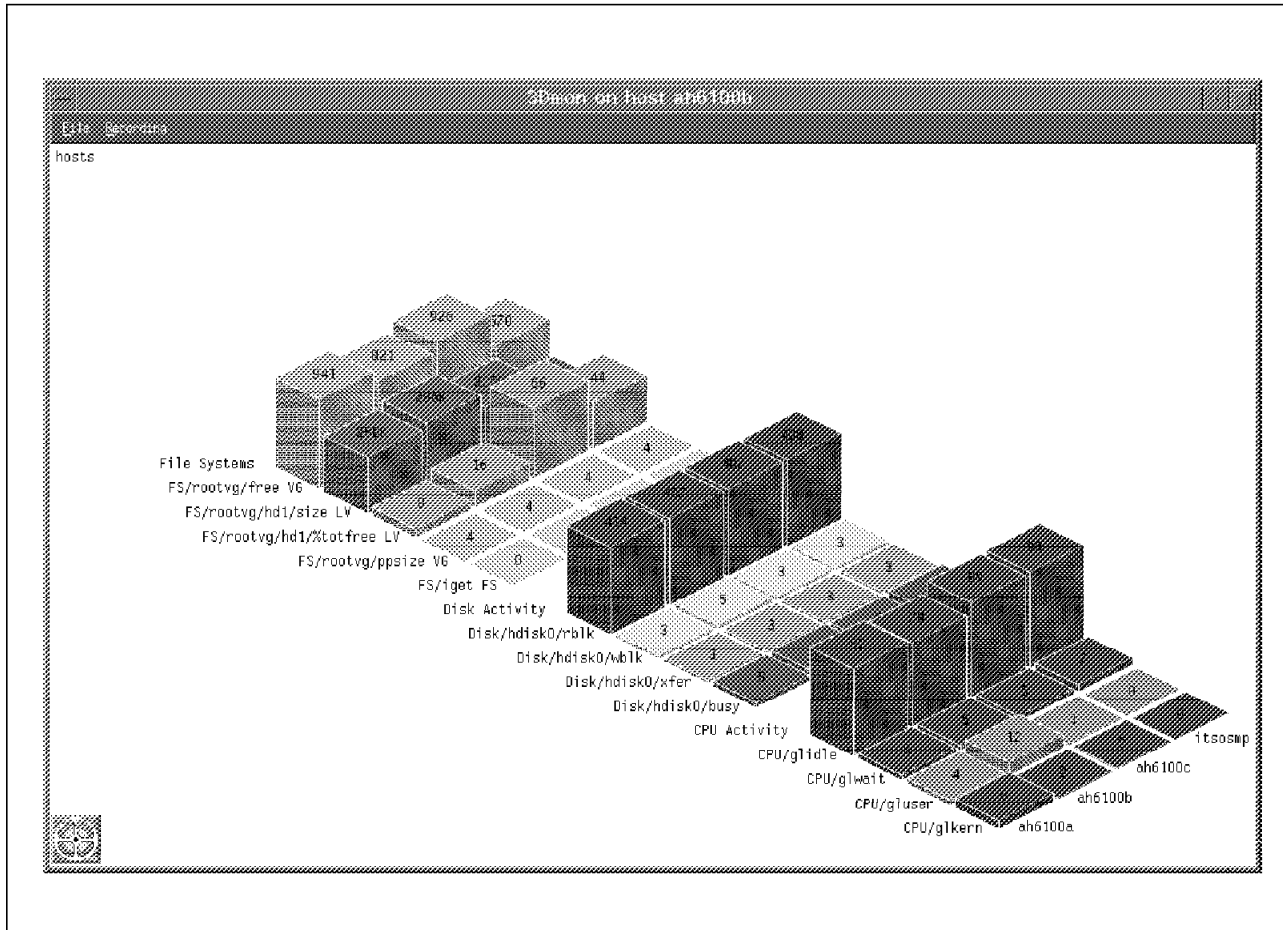


Figure 59. 3dmon - multi host Monitor

3.2 The 3dplay File

The 3dplay command is used for the playing back 3dmon recordings. When invoked, 3dplay will ask you for the file you would like to play back. To make your selection easier, files that match the filter R.3dmon are given as default choice. There are three ways to invoke 3dplay:

- From the command line - Example: Type in 3dplay.
- From the xmperv utilities - Example: Use the pull-down menu in xmperv.
- From 3dmon - Example: Use the pull-down in 3dmon and select **Playback**.

3.2.1 The 3Dplay Configuration File

A resource file is provided for 3dplay. The installation process installs the file in /usr/lib/X11/app-defaults. A copy also exists in /usr/samples/perfmgr as 3Dplay.resour.

3.2.2 The 3dplay.hlp Help File

The file containing help messages for 3dplay is 3dplay.hlp. The file resides in /usr/lpp/perfmgr as well as in the sample directory. This file may be changed the same way as in 3.3.3, "The exmon.hlp Help File" on page 70.

3.3 The Exmon Configuration

The exmon command works with the filtd daemon to generate exception packets that are defined in the filtd configuration file as alarms. The exmon command is used as a local and remote exception monitor. The file can be configured with timestamps, color coded or used for remote command execution.

3.3.1 The EXmon Resource File

The X Windows resource file for exmon defines resources you can use to enhance the appearance and behavior of exmon and is installed as /usr/lib/X11/app-defaults/EXmon. The following behavior can be modified with the exmon resource file.

Figure 60 shows the default configuration with the following code segment defining the font and foreground as well as background colors:

```
*GraphFont:      -ibm-block-medium-r-normal--15-100-100-100-c-70-iso8859-1
*DrawArea.background:  black
*DrawArea.foreground:  white
*Foreground:      black
*Background:      grey
```

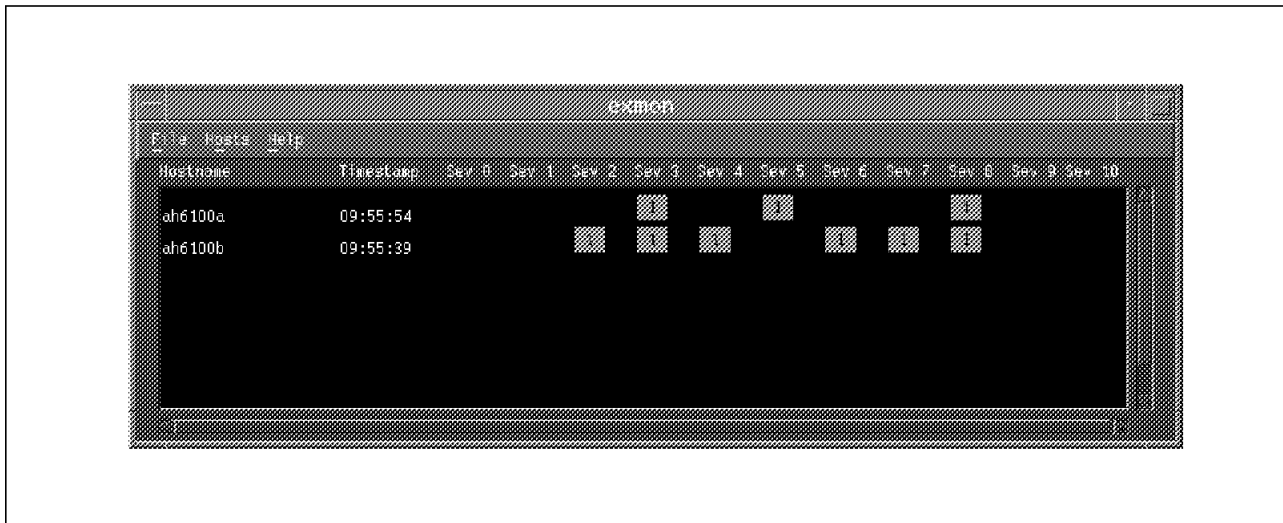


Figure 60. The Default exmon

In Figure 61 on page 67, the EXmon resource file has been modified by changing the foreground and background colors. The code for this figure is as follows:

```
*GraphFont:      -ibm-block-medium-r-normal--15-100-100-100-c-70-iso8859-1
*DrawArea.background:  white
*DrawArea.foreground:  black
*Foreground:      black
*Background:      white
```

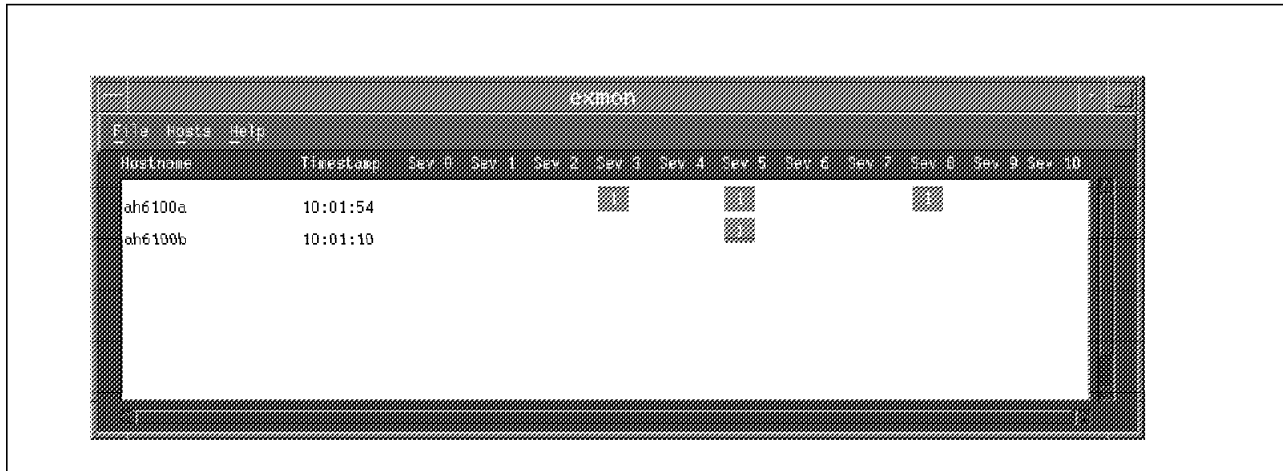


Figure 61. The Modified exmon

Note

The font may also be changed in EXmon. Please refer to 2.1.1, "The XMperf Font" on page 7, for the various font options available.

The cells are assigned a color depending on the coloring scheme specified in the resource file. Colors can be assigned based on the number of exceptions received in each cell, or a separate color can be assigned to each exception ID.

RangeDisplay may be set to true or false.

*RangeDisplay: true

The RangeColors may be changed from their defaults.

*RangeColor1: green
*RangeColor2: yellow
*RangeColor3: red

If RangeDisplay is true, it will use the following values:

(The 5 means that after the fifth exception, it will change color from the default green color to yellow).

*ValueRange1: 5

The 10 means that after the tenth exception the color will change from yellow to red and stay there for the rest of the exceptions.

*ValueRange2: 10

If RangeDisplay is false it will use the following colors. You may type over these colors with your desired colors:

*ValueColor0: blue
*ValueColor1: red
*ValueColor2: white
*ValueColor3: green
*ValueColor4: yellow
*ValueColor5: orange
*ValueColor6: grey
*ValueColor7: pink
*ValueColor8: magenta

```
*ValueColor9: GreenYellow
*ValueColor10: SkyBlue
```

The ExceptionText fields display what the user sees at the top of each exception.

```
*ExceptionText0: Sev 0
*ExceptionText1: Sev 1
*ExceptionText2: Sev 2
*ExceptionText3: Sev 3
*ExceptionText4: Sev 4
*ExceptionText5: Sev 5
*ExceptionText6: Sev 6
*ExceptionText7: Sev 7
*ExceptionText8: Sev 8
*ExceptionText9: Sev 9
*ExceptionText10: Sev 10
```

If your alarm file is configured to group similar exceptions with the same severity, you may wish to modify the severity tags. Refer to Figure 66 on page 87, for the output with these new exception names:

```
*ExceptionText0: CPU
*ExceptionText1: Page
*ExceptionText2: DISK
*ExceptionText3: FILE
*ExceptionText4: Sev 4
*ExceptionText5: Sev 5
*ExceptionText6: Sev 6
*ExceptionText7: Sev 7
*ExceptionText8: Sev 8
*ExceptionText9: Sev 9
*ExceptionText10: Sev 10
```

After making the above modifications in the EXmon resource file, exmon is now displayed as seen below:

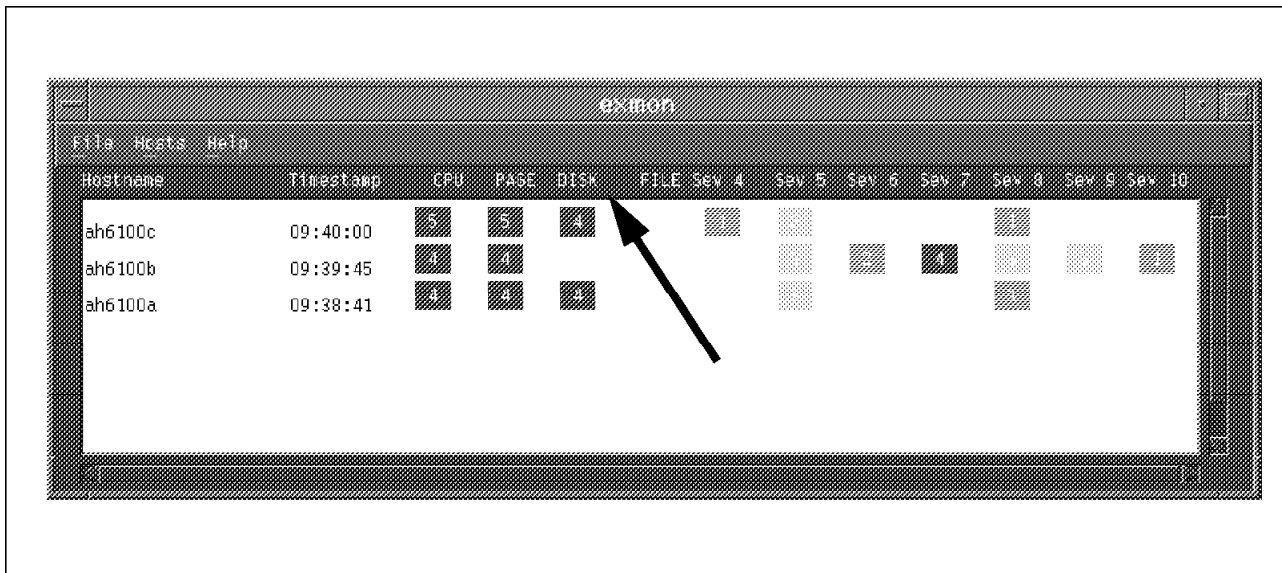


Figure 62. Modified EXmon

3.3.2 The exmon.cf Configuration File

In configuring the exmon.cf file, you are able to insert commands into the file so that when you click the left mouse button on a specific host, the command display is brought up for you.

Below is an extract from the exmon.cf file, and the code for some extra commands is added in bold:

```
#Display processes for host
3dmon Processes:3dmon -h%s -cProc &
#chmon for selected host
chmon:aixterm -n chmon -T chmon -e ksh -c "(sleep 1; chmon %s)" &
#Start xmperf on host
xmperf:xmperf -h%s &
#xmpeek statistics
xmpeek:aixterm -n xmpeek -T xmpeek -e ksh -c "(xmpeek %s; read)" &

# The following is an example of tailoring your exmon.cf file with
# your own commands

# Start 3dmon with the multi configuration set
3dmon multi:3dmon -h%s -cmulti &
# Start an aixterm pinging a specific host
ping:aixterm -n ping -T ping -e ksh -c "(ping %s; read)" &
# Telnet in to the specified host
telnet:aixterm -n telnet -T telnet -e ksh -c "(telnet %s; read)" &
# Check who is logged on to the host
who:aixterm -n who -T who -e ksh -c "(who -u; read)" &
# Check the local File Systems on the host
df:aixterm -n df -T df -e ksh -c "(df -k; read)" &
```

Note

The %s indicates to the program the hostname of the host the user is on.

After inserting the code, the following was output to the screen when the left mouse button was clicked while on the host:

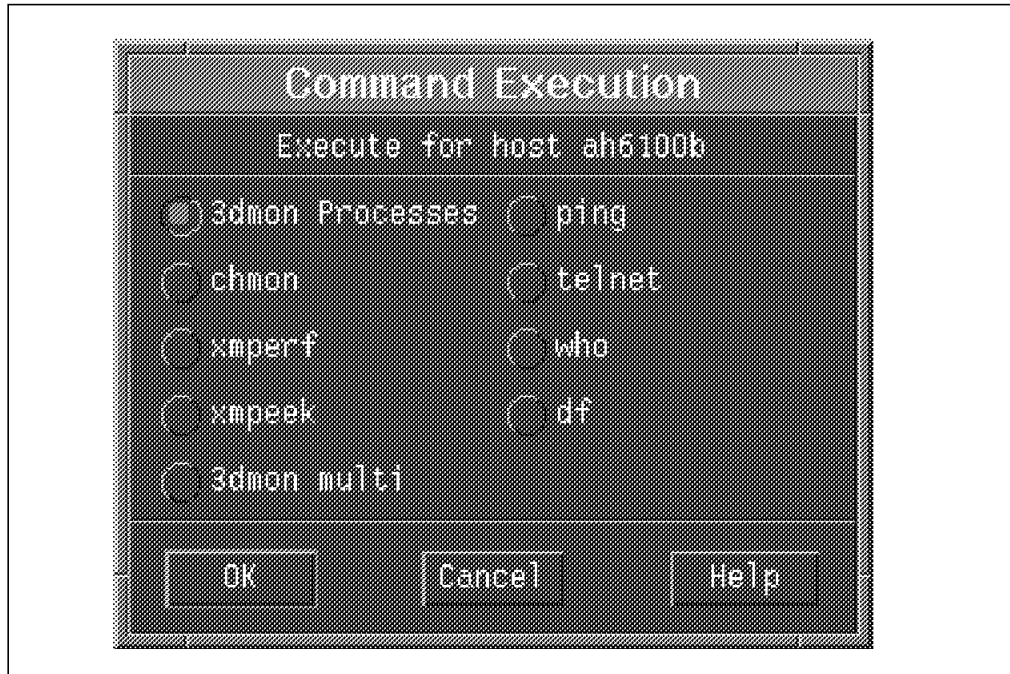


Figure 63. Command Options of exmon

3.3.3 The exmon.hlp Help File

This file is of the same format as the xmperf.cf file located in 2.4, “The xmperf.hlp Help File” on page 45. You have the selection of updating the original context of material or entering a new topic. In the following example some new data was inserted in the \$help: Command Execution section:

\$help: Command Execution

The exmon program gives the user the ability to execute commands for certain hosts. The user clicks the pointer on a hostname that is displayed in the exmon monitoring window. A dialog box is popped up that displays the commands the user may execute for that host. The user selects a command and then clicks the OK button.

| | |
|------------------------|--|
| 3dmon Processes | #Display processes for host |
| chmon | #chmon for selected host |
| xmperf | #Start xmperf on host |
| xmpeek | #xmpeek statistics |
| 3dmon multi | #Display 3dmon multi view(CPU,fs and disk Activity) |
| ping | #ping the host |
| telnet | #telnet to the host |
| who | #Who is logged on the host |
| df | #Check disk file systems on the host |

<more>

Note

The highlighted text was inserted into the original exmon.hlp file.

Below is the output when help is selected from within the command panel. It now shows the new text added into the exmon.hlp file:

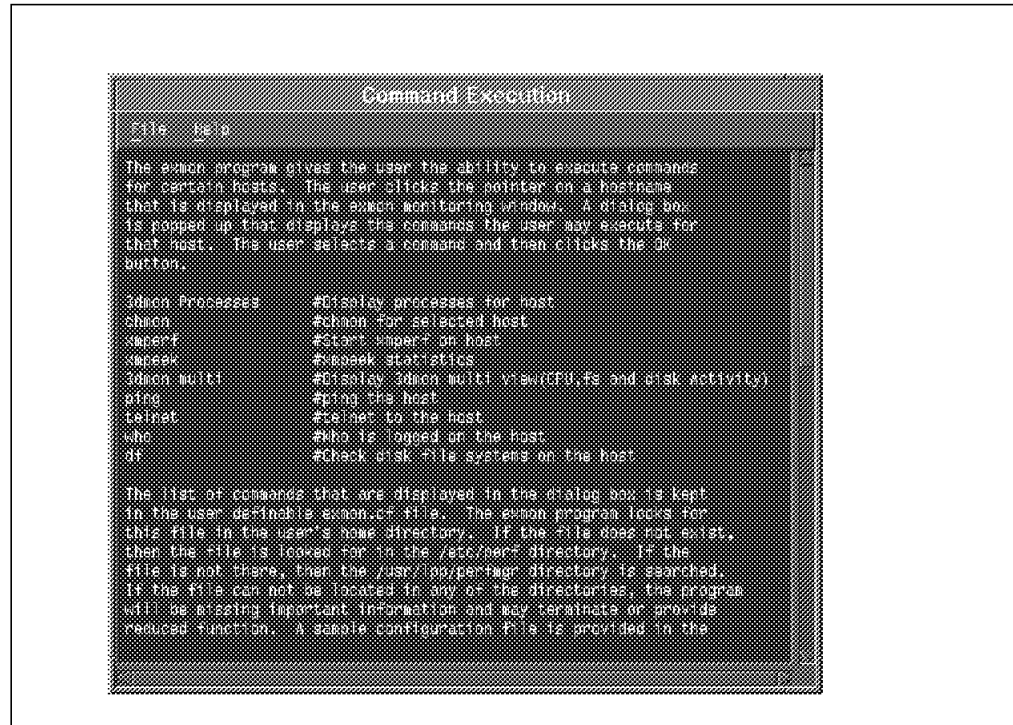


Figure 64. The exmon Main Help Panel

Chapter 4. Agent Configuration

The Performance Toolbox (PTX) agent component is a collection of programs that provide performance statistics across a network or locally.

Before editing the configuration files, it is strongly recommended that the user copy the default configuration files from `/usr/lpp/perfagent` into the `/etc/perf` or `$HOME` directory so the original defaults can be restored at any time. In this chapter the following files were copied into the `/etc/perf` directory and will be the files that we use:

- `xmservd.res` (resource file)
- `xmservd.cf` (configuration file)
- `filter.cf` (configuration file)

Note

There is no default `xmservd.cf` configuration file. One is created in 4.1.2, "The `xmservd.cf` File" on page 76.

4.1 The `xmservd` Configuration

The primary agent component is the data supplier, `xmservd`.

The `xmservd` daemon is designed to be started from the `inetd` daemon. Even when you start the daemon manually, it reschedules itself via `inetd` and lets the manually started process die.

The definition of `xmservd` in `/etc/inetd.conf` is done automatically during the installation process.

```
xmquery dgram udp wait root /usr/bin/xmservd xmservd -p3
```

There are additional command line options available to `xmservd`. Refer to the *Performance Toolbox for AIX: Guide and Reference, SC23-2526, Chapter 11* for a complete list of options. The following are the most commonly used options:

- l** Sets the *time_to_live* after feeding of statistics. The default is 15 minutes. Setting this value to zero will cause the daemon to remain active after all requests for data feeds have stopped. This value should be set to zero if alarms have been defined in `filter.cf`.
- p** Sets the trace level, which determines the types of events written to the log file `/etc/perf/xmservd.log1` or `/etc/perf/xmservd.log2`. Must be followed by a digit from 0 to 9, with 9 being the most detailed trace level. The default trace level is zero, which disables tracing and logging of events but logs error messages.
- x** Sets the execution priority of `xmservd`. Use this option if the default execution priority of `xmservd` (fixed priority of 24) is unsuitable in your environment. Generally, the daemon should be given as high execution priority as possible (a smaller number gives a higher execution priority.)

Here is an example:

Create a backup copy of /etc/inetd.conf and modify the xmservd startup definition to the following:

```
xmquery dgram udp wait root /usr/bin/xmservd xmservd -p3 -l0 -x59
```

After modifying the inetd.conf entry, you must refresh the daemon for inetd to recognize the changes.

```
refresh -s inetd
```

If you want the daemon to be started automatically as part of the boot process, you can add the following two lines at the end of the /etc/rc.tcpip file:

```
sleep 10          # only needed if using SNMP agents
/usr/bin/xmpeek   # kicks off the xmservd daemon
```

Like many other daemons, xmservd will interpret the receipt of the signal SIGHUP (kill -1) as a request to refresh itself.

The basic configuration files for xmservd control which systems have access to the data collection statistics, which dynamic data suppliers are started when xmservd is invoked, whether or not communication with SNMP clients can occur, and which statistics are recorded for later analysis.

When xmservd is started, the agent will search for the configuration files in the following order and use the first occurrence:

1. /etc/perf
2. /usr/lpp/perfagent (default install directory)

4.1.1 The xmservd.res File

It is possible to control which systems on a network have access to the statistics provided by xmservd. The xmservd.res file is the file that limits access to xmservd, starts any requested dynamic data suppliers at application start-up, and enables xmservd to communicate with SNMP clients.

4.1.1.1 Controlling Access

Access is controlled by the following stanzas:

- only:** When this stanza is present, access is restricted to only those hosts named after the stanza. Hostnames are separated by blanks, tabs or commas.
- always:** When this stanza is present, access is always granted to hosts that are named after the stanza. If the **only:** stanza is also used, and the hostname does not appear in that stanza, access will be denied before the **always:** stanza is checked.
- max:** This stanza must be followed by the maximum number of simultaneous data consumers to be allowed. Any hosts listed in the **always:** stanza will not be counted toward the **max:**.

Note

The colon is part of the stanza and must begin in column one of a line.

There may be more than one occurrence of each stanza, but in the case of **max:**, only the last occurrence is used.

```
only: mickey minnie pluto daisy donald
only: c3po r2d2 luke hans leah
always: luke leah
always: mickey minnie
max: 5
```

4.1.1.2 Invoking Dynamic Data Suppliers

Programs that supply `xmservd` with statistics are called Dynamic Data Suppliers (DDS). Before a DDS can start supplying statistics to `xmservd`, the DDS must register with `xmservd`. DDS programs can either be started manually or by another process when their presence is requested, or they can be started automatically when `xmservd` starts.

There is no limit to the number of DDS programs that can be started by `xmservd`. Each DDS must have a stanza:

supplier: The stanza includes the colon (:) and must be followed by at least one byte of white space and the full path name of the DDS program and any invocation flags or parameters.

```
supplier: /usr/bin/filtd -p5
supplier: /usr/lpp/perftools/progcntd
```

4.1.1.3 SNMP

The Simple Network Management Protocol (SNMP) is a network protocol based on the Internet protocol.

The objective of the Performance Toolbox is very different from that of most SNMP applications, in particular, IBM's NetView. SNMP network management programs are primarily concerned with supervision and corrective action aimed at keeping the network resources available. Performance Toolbox is concerned with continuous monitoring of the resource utilization.

Somewhere between NetView and PTX lies a gray area that both products are interested in. Thus, Performance Toolbox statistics may need to be available to NetView and other SNMP applications.

A common access mechanism is available through the `xmservd/SMUX` interface. The interface allows `xmservd` to present all its statistics to the SNMP agent (`snmpd`) as read-only variables. The `xmservd/SMUX` interface is invoked by placing a single stanza in the `xmservd.res` configuration file. The stanza must begin in column one of a line. The line must read:

```
dosmux
```

The `xmservd` daemon can produce a Management Information Base (MIB) file that describes all the variables currently exported to `snmpd`. To produce this file, a SIGINT (`kill -2`) must be sent to `xmservd`. When `xmservd` received a SIGINT, it will produce the file `/etc/perf/xmservd.mib`. Refer to the *Performance Toolbox for AIX: Guide and Reference*, SC23-2625, Chapter 13 for additional information on SNMP and MIBs.

4.1.1.4 Registering Changes to xmservd.res

After making any changes to xmservd.res, you will need to refresh the daemon.

Since refreshing the daemon may be necessary several times during the configuration process, a script to remove the log files and refresh the daemon is included.

```
#!/usr/bin/ksh
#
# This script will:
# Remove log files and Refresh xmservd
#
if [ `whoami` != "root" ]
then
    print "\n ==> You must be root to refresh xmservd.\n"
    exit -1
fi

if [ -f /etc/perf/xmservd.log* ] || [ -f /etc/perf/filter.log* ]
then
    rm -f /etc/perf/*log*
    print "\n ==> /etc/perf log files have been removed"
fi

oldpid=`ps -ef | grep xmservd | grep -v grep | grep -v $0 | awk '{print $2}' ~`
if [ -z "$oldpid" ]
then
    xmpeek > /dev/null
else
    kill -1 $oldpid
    print "\n ==> xmservd has been sent a SIGHUP PID=$oldpid"
fi
sleep 5
newpid=`ps -ef | grep xmservd | grep -v grep | grep -v $0 | awk '{print $2}' ~`
if [ -z "$newpid" ]
then
    print "\n ==> Unable to start xmservd, see /etc/perf/xmservd.log*"
elif [ "$newpid" -eq "$oldpid" ]
then
    print "\n ==> Unable to kill xmservd PID=$newpid\n"
else
    print "\n ==> xmservd has been restarted PID=$newpid\n"
fi
```

4.1.2 The xmservd.cf File

PTX provides the ability to record statistics for after-the-fact analysis of performance problems. This capability is called the xmservd recording facility and is controlled through the xmservd configuration file. The xmservd.cf file is the file that controls which performance statistics are recorded. There is no default file provided. You need to create your own file in /etc/perf.

If xmservd finds a valid configuration file in any of the default search locations, recording will be enabled.

The configuration file must contain:

- One retain line
- One frequency line

- One or more metric lines
- One or more start/stop lines

The configuration file may contain:

- One or more command lines
- One or more hot lines

4.1.2.1 Retain Line

The retain line indicates how many days worth of data to keep before `xmservd` removes the file. The retain line also indicates how many days worth of data to store in each file.

```
#retain days_to_keep [days_per_file]
#days_to_keep specifies the minimum number of days to keep a recording file
#before xmservd deletes it.
#days_per_file is optional, if not specified, defaults to days_to_keep
retain 14 2
```

4.1.2.2 Frequency Line

The frequency line indicates the default recording interval in milliseconds. To keep disk storage reasonable, it is recommended that you select an interval of 5 or more minutes.

```
#Frequency interval
#the interval is specified in milliseconds 60000=1 minute, 300000=5 minutes
frequency 300000
```

4.1.2.3 Metric Lines

The metric lines indicate which statistics you want recorded. If no interval is specified on the entry, the default interval is used. If creating this file from the output of `xmpeek -l`, you will need to remove the hostname and the forward slash to the left of the metric name as well as the comment information to the right of the metric name.

```
#metric [interval]
#the full pathname of the statistic
#and any interval if the default is not to be used
CPU/cpu0/user
CPU/cpu0/kern
CPU/cpu0/wait
Mem/Real/numfrb
Mem/Virt/pagein
```

Note

If the metric (statistic) does not exist on that system, `xmservd` will not be able to process the recording file.

4.1.2.4 Start Lines

Start lines indicate the start and stop times for recording. After the keyword `start`, the next field is the days to start monitoring. Sunday is day 0. The field can either be a single day, a range of days, or a comma separated list. The next field is the hour to start monitoring on a 24-hour clock. This field can also be a single number, a range of numbers, or a comma separated list. The next field is the minute you want recording to start. This field can be a single number or a

comma-separated list. The last three fields are the days, hours, and minutes to stop recording.

```
#start dd hh mm dd hh mm
#dd: 0=Sunday, 6=Saturday can be a single number or a range, e.i. 1-5
#hh: 24 hour clock with 00=midnight, can be a single number or a range
#mm: can be a single number or a series of comma separated minute values
start 1-5 8 30 1-5 17 0
```

4.1.2.5 Command Lines

One of the two new features added to the xmservd recording facility is the ability to execute one or more commands or scripts at the time an old recording file is deleted. These are specified in the configuration file with the following format:

```
command /bin/ptxmerge /var/perf/temp %s /var/perf/year_to_date
command /bin/mv -f /var/perf/temp /var/perf/year_to_data
```

The %s in the line refers to the file to delete. The first line uses the ptxmerge program to merge the recording file, which is about to be deleted with the year-to-date file of accumulated recording files, and place the output from the merge to a temporary file. The last line moves the temporary file over to the previous year-to-date file. Note that this series of commands is not safe; it is meant only to illustrate the facility. To do the above, you should use a script that makes sure that lack of disk space doesn't cause you to lose data.

4.1.2.6 Hot Lines

The second new feature added to the xmservd recording facility is the ability to record *HotSets*. HotSets permit the users to monitor metrics by activity rather than by name. The following is the layout of the lines to define HotSets. The values correspond to the arguments to the SpmiAddSetHot subroutine call:

```
# format of line to define hotfeed followed by examples:
#
# key                max thres   fre                seve trap
# word metric        resp hold   quency feed_type  except_type rity no
#
# hot LAN/*/framesin  1    0    60000 Always
# hot Disk/*/busy     3    50   10000 Threshold Trap    0  14
# hot FS/*/%totfree   3    95   300000 Threshold Both    4  16
# hot FS/rootvg/*/%totfree 0    95   300000 Always
# hot RTime/LAN/*/above99 3    80   300000 Threshold Exception 2
```

hot The key word indicating HotSet recording.

metric The metric with a wildcard in the specification.

maxresp The maximum number of responses to record. If the feed_type is Threshold, this value must be greater than 1. One exception/trap is sent for each metric that exceeds the threshold up-to the maximum value of this field.

threshold If feed_type is set as the Threshold, this field is the value which must be exceeded for the exception/trap to be sent. If the value is specified as a negative number, the threshold is considered to be exceeded if the monitored metric is lower than the numeric threshold value.

frequency The frequency to monitor the metrics. This value is in milliseconds.

feed_type The valid types are: Always or Threshold.

| | |
|-----------------------|--|
| exception_type | The valid types are: Exception, Trap or Both. |
| severity | If sending an exception, the severity level for the exception. |
| trap_number | If sending a trap, the trap number to send. |

Each line defines a separate HotSet.

4.1.3 The xmscheck Command

Once the xmservd.cf file is created, you need to make sure that it can be parsed by the daemon.

The xmscheck command, if invoked without any flags, will show if a configuration file is in effect for xmservd.

If xmscheck is invoked with a file name, a lengthy output is displayed to stdout. Any errors in the retain, frequency or metric sections are indicated in a text message after the error.

A table indicating the start/stop times is also displayed.

| | |
|---|---|
| 0 | Recording is inactive. |
| 1 | Recording is active. |
| 2 | Recording is inactive. A stop request was given for the minute. |
| 3 | Recording is active. A start request was given for the minute. |

Below is the output from running xmscheck:

```
Using xmservd.cf file in /etc/perf/xmservd.cf
+++++++ configuration records begin ++++++++
#
# XMSERVD Recording configuration file
#
# Keep files for at least x days
# let each file contain y days recordings
# retain x y
retain 14 2

# Set default sampling interval (1 minute=60000)
# frequency 60000
frequency 30000

# List stats to record at the default frequency
CPU/cpu0/user
CPU/cpu0/kern
CPU/cpu0/wait

Mem/Real/numfrb
Mem/Virt/pagein
Mem/Virt/pageout
Mem/Virt/pgspgin
Mem/Virt/pgspgout

PagSp/totalsize
PagSp/totalfree

Disk/hdisk0/busy
Disk/hdisk0/rblk
Disk/hdisk0/wblk

Proc/runque

IP/NetIF/lo0/ipacket
IP/NetIF/lo0/ioctet
IP/NetIF/lo0/opacket
IP/NetIF/lo0/octet
IP/NetIF/tr0/ipacket
IP/NetIF/tr0/ioctet
```

4.2 The filtd Configuration

Another agent component is `filtd`. The configuration file for `filtd` reduces data by processing previously defined expressions into new statistics. It also supplies `exmon` with exceptions that occur on the monitored system and defines alarms that trigger actions.

The `filtd` daemon has two purposes:

1. Provides newly defined statistics to the `xmservd` daemon
2. Triggers defined alarm actions

The most common options for `filtd` are:

- f** Override the default configuration file name. Must be followed by the name of the file containing the data reduction and alarm definitions.
- p** Specifies the level of detail to be written to the log file `/etc/perf/filter.log1` or `/etc/perf/filter.log2`. Must be followed by a digit from 0 to 9, with 9 being the most detailed trace level. The default detail level is 0.

The `filtd` command is designed to run as a daemon. It is typically started by `xmservd` from an entry in the `xmservd.res` file. There is an entry in the file that will need to be uncommented if you wish to start it automatically.

```
supplier:      /usr/bin/filtd -p5
```

4.2.1 The filter.cf Configuration File

The default search path for the configuration file `filter.cf` is:

1. `$HOME`
2. `/etc/perf`
3. `/usr/lpp/perfagent`

After making changes to `filter.cf`, you need to register the changes with `xmservd`. As root, you must kill `filtd` and then start it again with:

```
/usr/bin/filtd -p5 &
```

You may kill `filtd` by using the command `kill -15 (process number)`. To find the process number, use the `ps -eta | grep filtd` command.

Example:

```
ps -eta | grep filtd
```

gives you the following output if `filtd` is running:

```
33016      -  0:52 filtd
```

where 33016 is the process ID. You may then use:

```
kill -15 33016
```

to kill the process, then check that `filtd` has been killed:

```
ps -efa | grep filtd
```

When you get no response back, you are now able to restart `filtd`:

```
filtd -p5 &
```

Note

Each time you restart filtd, it will be assigned a different process number. After restarting filtd, you may have to wait until the process is running, this may take a minute to configure itself.

The filter.cf configuration file has two main purposes:

1. Define new statistics (data reduction)
2. Set alarms

Note

After making any changes to your filter.cf configuration file, if you have any errors on startup, they will be found in the files filter.log1 or filter.log2 located in your /etc/perf directory.

4.2.1.1 Defining New Statistics

Let's define a new statistic, cpubusy, which consists of CPU user + kern. The following code was inserted into the filter.cf configuration file:

```
cpubusy = CPU_cpu0_user + CPU_cpu0_kern "Cpu busy (user+kernel)"
```

Note

The syntax of statistics in the filter.cf file is different from all other components of the Performance Toolbox. Instead of a forward slash (/) separating the various hierarchy levels, filter.cf uses an underscore (_).

The created statistic is defined as:

```
DDS/IBM/Filters/cpubusy
```

To check that the update was successful, please refer to "Verifying A New Statistic" on page 83.

Using Wildcards: When creating new statistics, you can use wildcards as detailed below:

- + All values matching the wildcard are added together.
- * All values matching the wildcard are multiplied with each other. Unless all values are non-zero, the results will be zero.
- # Evaluates to a constant, which is the number of values that match the wildcard.
- > Evaluates to the maximum value of all those matching the wildcard.
- < Evaluates to the minimum value of all those matching the wildcard.

Example:

```
diskmax = Disk_>_busy "Busy percent - most busy disk"  
diskmin = Disk_<_busy "Busy percent - least busy disk"
```

The new statistic, `diskmax`, has the value of the disk busy percentage of the most active disk at any given time. Likewise, `diskmin` has the value of the disk busy percentage of the least busy disk.

You may now use these statistics in the `filter.cf` configuration for alarms. See 4.2.1.2, "Defining Alarms." The statistics are also available through using the `xmperf` command. For more information on this please refer to the redbook, titled *RS/6000 Performance Tools in Focus*, SG24-4989, Chapter 5.

Verifying A New Statistic: To check for the statistics, you may use the `xmpeek -l` command. You will see the following at the bottom:

```
/ah6100b/DDS/IBM/Filters/           Filters defined by DDS
/ah6100b/DDS/IBM/Filters/cpubusy     Cpu busy (user+kernel)
/ah6100b/DDS/IBM/Filters/diskmax     Busy percent - most busy disk
/ah6100b/DDS/IBM/Filters/diskmin     Busy percent - least busy disk
```

4.2.1.2 Defining Alarms

The purpose of alarms in the `filter.cf` configuration file is to:

- Invoke a command
- Send an Exception to `exmon`
- Pass a trap to SNMP

An alarm consists of an action part that describes what action to trigger and a conditional statement that defines the condition for triggering the alarm. Alarms must have the following format:

```
alarm_name: [command]{TRAPxxx}{EXCEPTION}condition DURATION FREQUENCY SEVERITY
```

Let's look at an example for free page space:

```
@pagfree: [aixterm -e ksh -c "banner $(hostname) PAGE LOW;read"] {EXCEPTION}
          PagSp_%totalfree < 1024 DURATION 60 FREQUENCY 15 SEVERITY 1 \
          "Test for Page free < 1024KB"
```

| | |
|--------------------|--|
| alarm_name: | Must start with an @ and otherwise contains only alphanumeric characters. @pagfree: |
| command | Commands enclosed in square brackets are executed when the condition is met. This could be as simple as a mail message sent to a system administrator. Be sure to use the full path name for any executables. [aixterm -e ksh -c "banner \$(hostname) PAGE LOW;read"] |
| TRAPxxx | If <code>xmservd</code> is configured to export statistics to <code>snmpd</code> , this action will pass the trap number <code>xxx</code> to <code>snmpd</code> . TRAP must be in all capital letters and followed by one or more decimal digits defining the trap number. |
| EXCEPTION | This action causes <code>filtd</code> to inform <code>xmservd</code> each time the condition is met. The <code>exmon</code> command can then be used to monitor the exceptions. {EXCEPTION} |
| condition | A boolean expression defining the alarm condition. PagSp_%totalfree < 1024 |

| | |
|------------------|---|
| DURATION | How long a condition must be met before the alarm is triggered. Duration is in seconds, default=1. DURATION 60 |
| FREQUENCY | The amount of time elapsed since the last time this same alarm was triggered. Frequency is in minutes, default=1. FREQUENCY 15 |
| SEVERITY | A numeric tag that can be associated with an alarm. Severity can currently be specified as a value from 0 to 10. The severity numbers are useful when running exmon to group similar alarms in the same category. SEVERITY 1 |

Invoking a Command: The following is a segment of code that outputs to the local user's screen a message stating the /tmp file system is full:

```
@tmpful: [aixterm -bg white -fg red -e ksh -c "(banner $(hostname) /tmp
FULL ; read)"] {EXCEPTION} FS_rootvg_hd3_%totused > 95 DURATION 60 \
FREQUENCY 15 SEVERITY 3 "/tmp > 95"
```

Here is a breakup of the above code:

@tmpfull Defines the alarm name to be tmpfull.

```
[aixterm -bg white -fg red -e ksh -c "(banner $(hostname) /tmp
FULL ; read)"]
```

- aixterm : Start an aixterm.
- -bg white -fg red : White background and red foreground for the aixterm.
- -e ksh -c : Execute korn shell command.
- "(banner \$(hostname) /tmp FULL; read)"
 - banner : Uses the banner command.
 - \$(hostname) : Outputs the hostname.
 - /tmp FULL : Outputs the /tmp filesystem is full.
 - : Defines a new line.
 - read : Waits for the user to press Enter or close the screen.

{EXCEPTION} Sends this exception to exmon.

FS_rootvg_hd3_%totused > 95 Defines the statistic to monitor and checks for the condition; if the condition is true, then action is triggered.

DURATION 60 The condition must be met for at least 60 seconds before being triggered.

FREQUENCY 15 The program waits 15 minutes between alarms.

SEVERITY 3 Determines the position of the exception in exmon.

"/tmp > 95" Comment.

When the alarm conditions are satisfied, the following will be output to the screen:

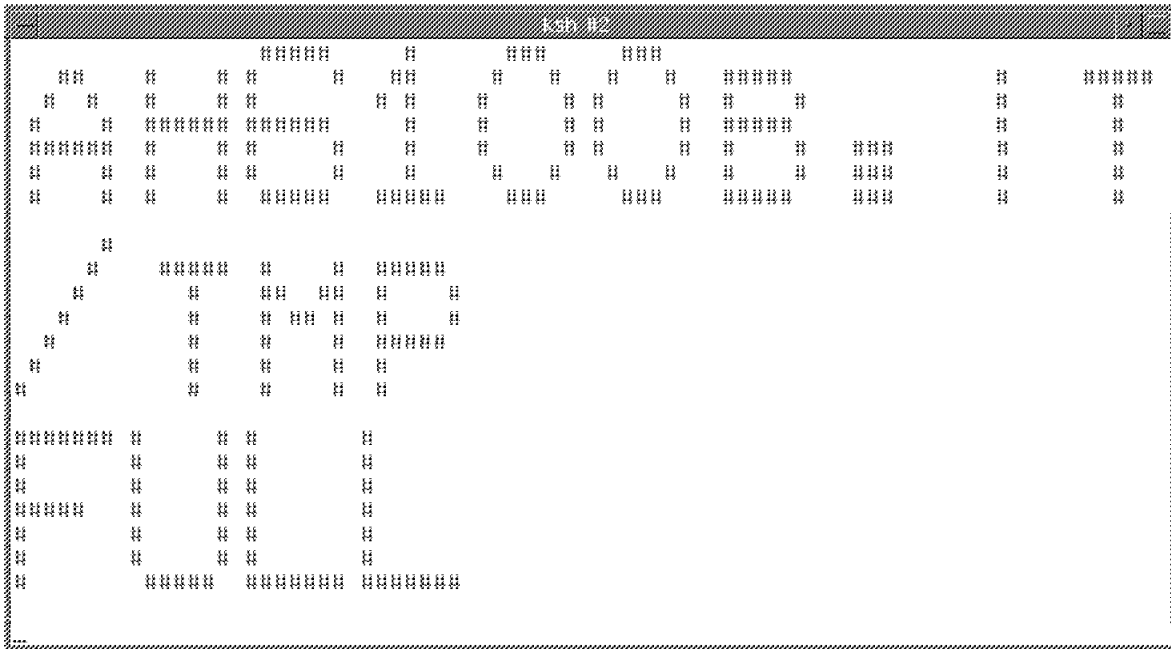


Figure 65. The filter.cf Command Output

Other useful commands to invoke could be:

- Run a vmstat command
[aixterm -e ksh -c "(vmstat 1 10;read)"]
- Mail a command executed to root:
[vmstat 1 60 | mail -s "CPU busy" root]
- Increase the filesystem size that you are monitoring (for example /tmp):
chfs -a size=+1 /tmp (where tmp is the file system)

In the example below, the segment of code was inserted into the filter.cf configuration file and saved; then the filtd daemon was stopped and restarted as detailed in 4.2, "The filtd Configuration" on page 81:

Example:

```
@cpubusy: [aixterm -e ksh -c "banner $(hostname) CPU BUSY;read"]{EXCEPTION}
DDS_IBM_Filters_cpubusy < 90 DURATION 60 FREQUENCY 1 SEVERITY 0 \
"Test if CPU busy > 90%"
@pagfree: [aixterm -e ksh -c "banner $(hostname) PAGE LOW;read"] {EXCEPTION} \
PagSp_totalfree > 256 DURATION 60 FREQUENCY 1 SEVERITY 1 \
"Test for Page free < 1024KB"
@diskbsy: [aixterm -e ksh -c "banner $(hostname) DISK BUS;read"] {EXCEPTION} \
Disk_hdisk0_busy < 35 DURATION 60 FREQUENCY 1 SEVERITY 2 \
"Test for busy disk"
@tmpful: [aixterm -bg white -fg red -e ksh -c "(banner $(hostname) /tmp \
FULL ; read)"] {EXCEPTION} FS_rootvg_hd3_%totused > 95 DURATION 60 \
FREQUENCY 15 SEVERITY 3 "/tmp > 95"
@cputrapp: [vmstat 1 60 | mail -s "CPU busy" root] DDS_IBM_Filters_cpubusy > 90 \
{TRAP33} {EXCEPTION} DURATION 300 FREQUENCY 120 SEVERITY 4 "CPU Busy"
```

Note

If you wish to output the display to another host on your network, then all you have to do is specify `aixterm -display HOSTNAME:0`, and the display is output to the hostname specified.

Sending an Exception to exmon: The following segment of code monitors the CPU usage of the system and uses the DDS defined statistic `cpubusy` defined in 4.2.1.1, "Defining New Statistics" on page 82.

```
@cpubusy: [aixterm -e ksh -c "banner $(hostname) CPU BUSY;read"]{EXCEPTION} \  
          DDS_IBM_Filters_cpubusy > 90 DURATION 60 FREQUENCY 15 SEVERITY 0 \  
          "Test if CPU busy > 90%"
```

Here is a breakup of what this code executes:

@cpubusy Defines the title of the alarm.

[**aixterm -e ksh -c "banner \$(hostname) CPU BUSY;read"**]

- **aixterm** : Creates a new aixterm.
- **-e ksh -c** : Executes a korn shell command.
- **"banner \$(hostname) CPU BUSY;read"**
 - **banner** : Writing is in banner form.
 - **\$(hostname)** : Uses the machine hostname and output to screen.
 - **CPU BUSY** : Outputs CPU BUSY to screen.
 - **read** : Waits for the user to press Enter or close the window.

{**EXCEPTION**} Will log the alarm in exmon.

DDS_IBM_Filters_cpubusy > 90 The trigger for the alarm.

DURATION 60 Checks every 60 seconds.

FREQUENCY 15 Outputs to screen every 15 minutes.

SEVERITY 0 Defines to exmon what severity the alarm is.

"Test if CPU busy > 90%" Comment.

When the alarm condition is satisfied and exmon is running, the following is output to the exmon screen depending on how long and how many exceptions have been recorded. In this example 16 exceptions (**1**) have been recorded:

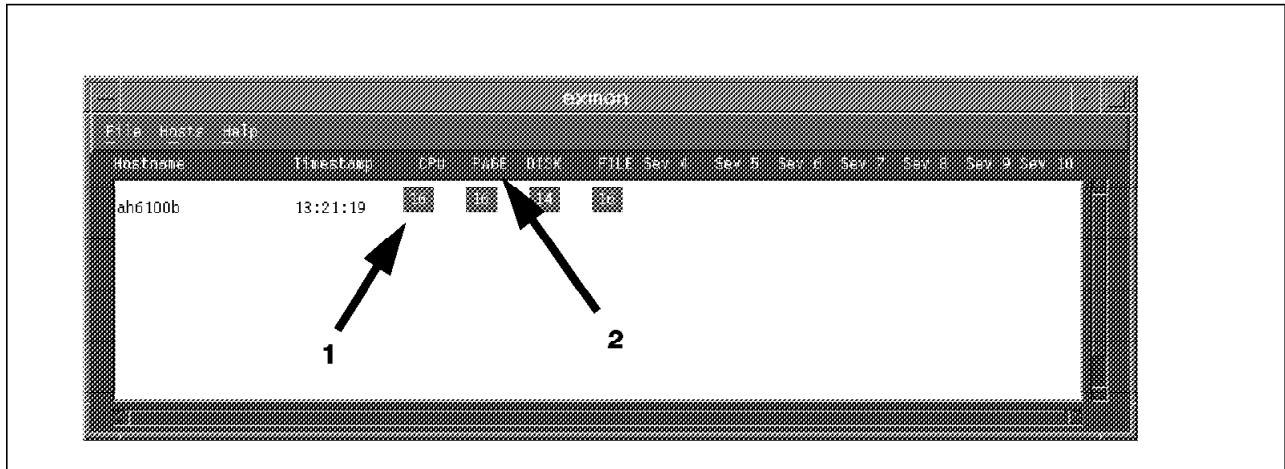


Figure 66. The exmon Output from the cpubusy Alarm

For tailoring exmon to read the CPU PAGE DISK FILE **(2)** instead of Sev 1, Sev 2, Sev 3, or Sev 4, please refer to 3.3.1, “The EXmon Resource File” on page 66.

Sample Exceptions: Below is a sample collection of alarms that will create many exceptions. Its purpose is just for demonstration, so you can get started with exmon. None of the alarm thresholds are reasonable.

```
@pswitch:{EXCEPTION} Proc_pswitch > 5000
    DURATION 1 FREQUENCY 1 SEVERITY 0 "Context Switches > 5000"
@cpu:{EXCEPTION} CPU_gluser + CPU_glkern > 50 \
    DURATION 10 FREQUENCY 5 SEVERITY 1 "CPU Usage Alarm"
@disk:{EXCEPTION}((Disk+_busy) / (Disk#_busy)) > 5 \
    || DDS_IBM_Filters_diskmax > 10 \
    DURATION 3 FREQUENCY 1 SEVERITY 2 "Disk Usage Alarm"
@runqueue:{EXCEPTION} Proc_runque > 1 \
    DURATION 1 FREQUENCY 1 SEVERITY 3 "Run Queue Alarm"
@swpqueue:{EXCEPTION} Proc_swpque > 1 \
    DURATION 1 FREQUENCY 1 SEVERITY 4 "Swap Queue Alarm"
@mem:{EXCEPTION} Mem_Real_%free < 20 \
    DURATION 1 FREQUENCY 3 SEVERITY 5 "Memory Alarm"
@pgsppgout:{EXCEPTION} Mem_Virt_pgspgout > 1 \
    DURATION 1 FREQUENCY 2 SEVERITY 6 "Page Space Page Out Alarm"
@pgsppgin:{EXCEPTION} Mem_Virt_pgsppgin > 1 \
    DURATION 1 FREQUENCY 2 SEVERITY 7 "Page Space Page In Alarm"
@syscallread:{EXCEPTION} Syscall_read > 25 \
    DURATION 1 FREQUENCY 2 SEVERITY 8 "System Calls: Reads > 25"
@syscallwrite:{EXCEPTION} Syscall_write > 25 \
    DURATION 1 FREQUENCY 1 SEVERITY 9 "System Calls: Writes > 25"
@FS:{EXCEPTION} FS_namei > 100 \
    DURATION 1 FREQUENCY 1 SEVERITY 10 "namei > 100"
```

Passing a Trap to SNMP: If you wish to use the SNMP function of PTX, then you must specify a TRAP followed by the trap number. The xmservd daemon will notice this trap and then send it on to the SNMP agent. This will only work if you specified in your xmservd.res file the option dosmux as detailed in 4.1.1.3, “SNMP” on page 75.

Example: CPU TRAP

```
@cputrap: [vmstat 1 60 | mail -s "CPU busy" root]
DDS_IBM_Filters_cpubusy > 90 \
{TRAP33} {EXCEPTION} DURATION 300 FREQUENCY 120 SEVERITY 4 "CPU Busy"
```

Here is a breakup of the above code:

@cputrap Defines the title of the alarm.

[vmstat 1 60 | mail -s "CPU busy" root]

- **vmstat 1 60** : Runs the vmstat command every second for sixty times.
- **mail -s "CPU busy" root** : mails the vmstat command executed to root and gives the subject field as CPU busy.

DDS_IBM_Filters_cpubusy > 90 The filter defined to check if the combined user and kernel CPU is over 90 percent. Refer to 4.1.1.2, "Invoking Dynamic Data Suppliers" on page 75, for more information on this filter.

{TRAP33} Sends a TRAP33 to the SNMP agent.

{EXCEPTION} Records this exception in exmon.

DURATION 300 This alarm has to be set off for over 300 seconds to trigger a response.

FREQUENCY 120 The last alarm that was triggered must have been at least 120 minutes old for this to trigger again.

SEVERITY 3 Notifies exmon that this alarm condition is a severity 3.

"CPU Busy" Comment on the alarm.

4.2.2 Creating Meaningful Alarms

To create meaningful alarms, the system administrator must first have a good understanding of AIX performance, the uses of the performance commands for bottleneck determination, and performance tuning strategies and tools. The system administrator must also have an understanding of the applications that run on the system and how they interact with the operating system. Unfortunately, the knowledge needed is not something you just read in a book to become an expert. Performance analysis and tuning takes book knowledge and a lot of hands-on experience.

Some general threshold values may be used to create alarms:

- CPU activity > 90 percent
- Disk busy >40 percent
- Filesystem > 90 percent used
- Page Space free < 1024 KB

After determining threshold values that are appropriate for your systems, you need to decide what actions to take if the conditions apply. Options include:

- Send an exception to be monitored by exmon.
- Send a mail message to a user.
- Execute a command for further information regarding the condition.

Performance Analyzer/6000: Performance Analyzer/6000 for AIX Systems is an offering available to customers through IBM Global Services. Performance Analyzer uses the data monitoring facilities of xmservd and alarm facilities of filtd to analyze your system's performance. The Performance Analyzer installation process creates several alarm definitions. When one of the alarms is triggered, the Performance Analyzer pa6000 command is invoked. The pa6000

command sends a mail message to a specified user stating that a problem has been detected. Performance Analyzer then provides a list of suggestions on how to alleviate the performance problem. Performance Analyzer may run additional performance tools to further detail the performance problems and suggest corrective action.

When a customer contracts the Performance Analyzer/6000, they not only get the tool but also assistance from a Services Specialist on customizing the alarms to meet their needs.

For additional information, contact IBM Global Services Inside Sales at 1-888-426-4343, or your IBM sales representative.

4.3 The SpmiResp Daemon

Measuring IP response times is done by the SpmiResp daemon. If the daemon is not running when the System Performance Measurement Interface (SMPI) receives a request for IP response time measurements, it is started by the Spmi library code. The daemon will remain active until either:

- It has been the only user of the Spmi interface for 60 seconds.
- No data consumer has requested response time data for 300 seconds.

4.3.1 The Resptime.cf File

The daemon looks for a configuration file in /etc/perf/Resptime.cf. The following values may be modified in the configuration file:

| | |
|-----------------|--|
| Interval | When running, the SpmiResp daemon is controlled by an interval timer loop. The default is to interrupt the daemon every 10 seconds. |
| Maxrate | The maximum rate SpmiResp will send Internet Control Management Protocol (ICMP) packets with packets per second. The default is 10 packets per second. |
| Weight | The weight a previous value has in finding the weighted average of the response time. The default is 75 percent. |

Note

The keyword must appear in column one of a line and white space must separate the keyword and the value.

If no configuration file is found, the default values are used.

4.3.2 The iphosts Command

Unlike other statistics, the contexts to monitor are not available until an application creates them. The iphosts command is a sample Data Consumer program that makes sure all IP response time contexts are created.

The iphosts command is designed to be started by xmservd from a supplier: entry in the xmservd.res file.

```
iphosts [-i] [-f filename] [-h "list of hostnames"]
```

-i Get hostnames by inviting data suppliers.

- f Get hostnames from specified file.
- h Get hostnames from command line.

Note

You must have root authority if invoking iphosts from the command line.

Because IP response time measurement uses the Internet Control Management Protocol (ICMP) to send responses to selected hosts, the hosts you want to monitor do not need to run the xmservd daemon. All that is required is that they can respond properly to ICMP echo requests. Because of this, response time to any node that talks ICMP, including dedicated routers and gateways, can be monitored.

4.4 The SpmiArmd Daemon

Measuring of Application Response Measurement (ARM) is done by the SpmiArmd daemon. Application Response Time can only be monitored if the applications have been instrumented. Refer to *Performance Toolbox for AIX, Guide and Reference*, SC23-2526, Chapter 15 for complete details.

4.4.1 The SpmiArmd.cf File

The daemon looks for a configuration file in `/etc/perf/SpmiArmd.cf`. The following values may be modified in the configuration file:

- Interval** When running, the SpmiResp daemon is controlled by an interval timer loop. The default is to interrupt the daemon every second.
- Weight** The weight a previous value has in finding the weighted average of the response time. The default is 75 percent.
- Timeout** The number of seconds the daemon should live with no activity occurring. The default is 900 seconds. A value of zero causes the daemon to live forever.

Note

The keyword must appear in column one of a line, and white space must separate the keyword and the value.

If no configuration file is found, the default values are used.

4.4.2 The ARM Libraries

The ARM specifications are shipped as a shared library such as `libarm.a` or `libarm.so`. Replacing the library with another library with the same interfaces will redirect application subroutines to the library last installed. The PTX implementation follows the above specifications, but also allows a customer installation to invoke both an existing ARM library and the PTX implementation of ARM.

Two libraries are shipped with the agent code:

- /usr/lib/libarm.a** A plain ARM implementation that does not invoke any preexisting ARM implementations.

/usr/lib/libarm2.a A replacement library.

To use the replacement library, convert the preexisting ARM library by running the following command as root:

```
armtoleg /usr/lib/libarm.a /usr/lib/libarmrepl.a > /dev/null
```

Then copy `/usr/lib/libarm2.a` to `/usr/lib/libarm.a`. The replacement library invokes the ARM functions in `libarmrepl.a` before invoking the PTX ARM implementation in the replacement library.

Both libraries depend on the `SpmiArmd` daemon to be started as root.

4.4.3 Runtime Control

The use of two environment variables allows the application to turn both levels of ARM instrumentation on or off.

INVOKE_ARM Controls the PTX ARM instrumentation. If the variable is not defined, or has any value other than `false`, PTX ARM instrumentation is active. If the replacement library is not used, the effect of setting this environment variable to `false` is that the PTX ARM library will function as a no-operation library and return zero on all calls.

INVOKE_ARMPREV Controls any ARM instrumentation that can be invoked from the PTX implementation through the replacement library. If the variable is not defined, or has any value other than `false`, the preexisting ARM instrumentation will be invoked, regardless of whether the PTX ARM instrumentation is active. If the replacement library is not used, this environment variable has no effect.

NOTE

If both variables are set to `false`, either PTX ARM library will function as a no-operation library or return zero on all calls.

4.5 The `xmpeek` Command

The `xmpeek` command has several functions:

- Starting `xmservd`.
- Displaying the status of `xmservd` for the local or a remote host.
- Listing all available statistics to monitor for a specific host.

Invoking `xmpeek` without any flags starts `xmservd` on the local host if it is not already executing. Whether or not the daemon was started, `xmpeek` displays several lines of status information. If no data consumer programs are utilizing the data feed from that host, a single line of information is returned. Otherwise, information regarding which systems have instruments or are utilizing the data feed are displayed.

```

Statistics for xmservd daemon on *** ah6100a ***

  Statsets currently defined:    12
  Statsets currently active:    12
  Hotsets currently defined:    0
  Hotsets currently active:    0
  Remote monitors currently known: 12

```

| Type | --No of sets---- | Values | Packets | Internet Address | Port | Hostname |
|------|------------------|--------|---------|------------------|------|----------|
| | Defined | Active | Sent | | | |
| Stat | 8 | 8 | 39 | 9.3.1.70 | 1345 | ah6100a |
| Stat | 1 | 1 | 23 | 9.3.1.71 | 2474 | ah6100b |
| Stat | 1 | 1 | 23 | 9.3.1.71 | 2576 | ah6100b |
| Stat | 1 | 1 | 13 | 9.3.1.70 | 3173 | ah6100a |
| Stat | 12 | 12 | 98 | | | |
| Hot | 0 | 0 | 0 | | | |

Figure 67. Output from xmpeek

If invoked with the `-l` flag, a list of statistics that can be monitored on the local system are displayed. Likewise, if the `-l` is followed by a hostname, the statistics that can be monitored on that host are displayed. Since the output is lengthy, it should be redirected to a file.

| | |
|--|---|
| /ah6100a/CPU/ | Central processor statistics |
| /ah6100a/CPU/gluser (percent) | System-wide time executing in user mode |
| /ah6100a/CPU/glkern mode (percent) | System-wide time executing in kernel |
| /ah6100a/CPU/glwait | System-wide time waiting for IO (percent) |
| /ah6100a/CPU/glidle | System-wide time CPU is idle (percent) |
| /ah6100a/CPU/gluticks user mode | System-wide CPU ticks executing in |
| /ah6100a/CPU/glkticks kernel mode | System-wide CPU ticks executing in |
| /ah6100a/CPU/glwticks | System-wide CPU ticks waiting for IO |
| /ah6100a/CPU/gliticks | System-wide CPU ticks while CPU is idle |
| /ah6100a/CPU/cpu0/ | Statistics for processor # 0 |
| /ah6100a/CPU/cpu0/user (percent) | Time executing in user mode(percent) |
| /ah6100a/CPU/cpu0/kern (percent) | Time executing in kernel mode |
| /ah6100a/CPU/cpu0/wait | Time waiting for IO (percent) |
| /ah6100a/CPU/cpu0/idle | Time CPU is idle (percent) |
| /ah6100a/CPU/cpu0/uticks | CPU ticks executing in user mode |
| /ah6100a/CPU/cpu0/kticks | CPU ticks executing in kernel mode |
| /ah6100a/CPU/cpu0/wticks | CPU ticks waiting for IO |
| /ah6100a/CPU/cpu0/iticks | CPU ticks while CPU is idle |
| /ah6100a/CPU/cpu0/pswitch processor | Process context switches on this |
| /ah6100a/CPU/cpu0/syscall | Total system calls on this processor |
| /ah6100a/CPU/cpu0/read | Read system calls on this processor |
| /ah6100a/CPU/cpu0/write | Write system calls on this processor |
| /ah6100a/CPU/cpu0/fork | Fork system calls on this processor |
| /ah6100a/CPU/cpu0/exec | Exec system calls on this processor |

Figure 68. Partial Output from xmpeek -l

Chapter 5. Recording and Playback

If you will need recorded information and a playback facilities, then some tools are included with the manager code (xmperf, 3dmon, azizo, ptxrlog). Others are packaged with the agent code (xmservd, ptxtab, ptxsplit).

5.1 Recording

Recording with PTX can be executed in many different ways. Here are a few common ways to do it with:

- xmperf
- 3dmon
- ptxrlog
- xmservd

5.1.1 Recording with xmperf

Recording of statistics can be initiated for one or more instruments in a console or for all instruments in a console. Recording can be active for more than one console at a time.

All recordings from any one console always go to a file in the \$HOME/XmRec directory, with a name of R. followed by the name of the console. For example, the recording file for a console named "Remote CPU Stats" would be \$HOME/XmRec/R.RemoteCPUStats.

To record only a specific instrument, click on the instrument. You should see a dashed line surrounding the instrument.

From the console menu bar, select **Recording**. You then have two choices:

| | |
|-----------------------------|---|
| Console Recording | To record all instruments in the console. |
| Instrument Recording | To record just a highlighted instrument. If this selection is grayed out, you must first select the instrument to record. |

From the **Recording** menu bar pull-down, the **Console Recording** and **Instrument Recording** gives you five choices:

| | |
|-----------------------------------|--|
| Save Buffer | This saves all statistics viewable in the instrument's history window since the console was opened. |
| Begin Recording | This begins recording statistics until End Recording is selected. |
| Save & Begin Recording | This saves all statistics in the history window and begins recording until End Recording is selected. |
| End Recording | This item is grayed out unless recording has been started. It is also used to stop the recording. |
| Annotation | This item be grayed out unless recording has been started. It is also used to add timestamps into the recording with a note. |

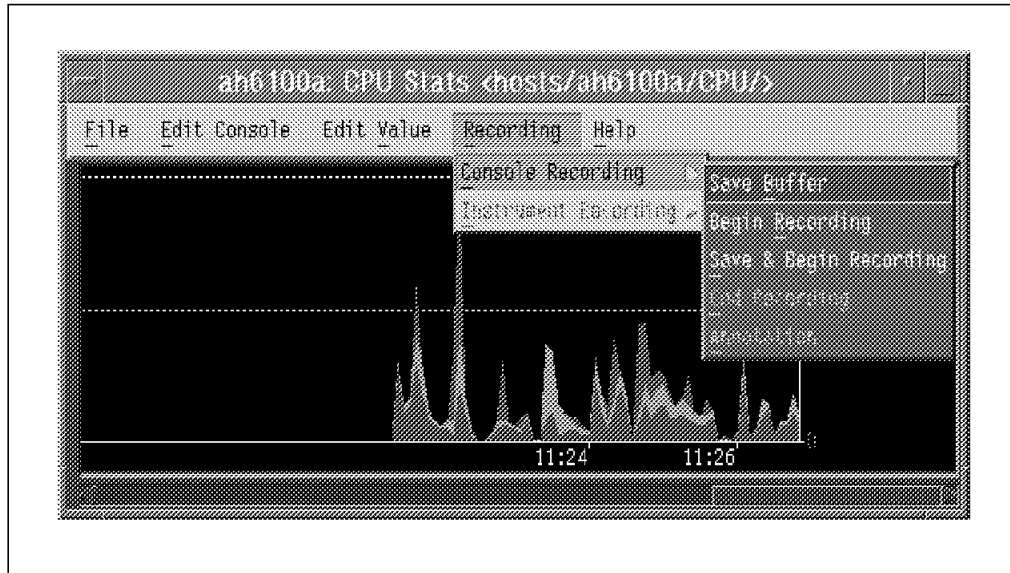


Figure 69. Example of Recording Pull-down Menu

If a recording file exists from a previous session, a pop-up window will appear prompting whether to Append, Replace, Cancel, or request Help.

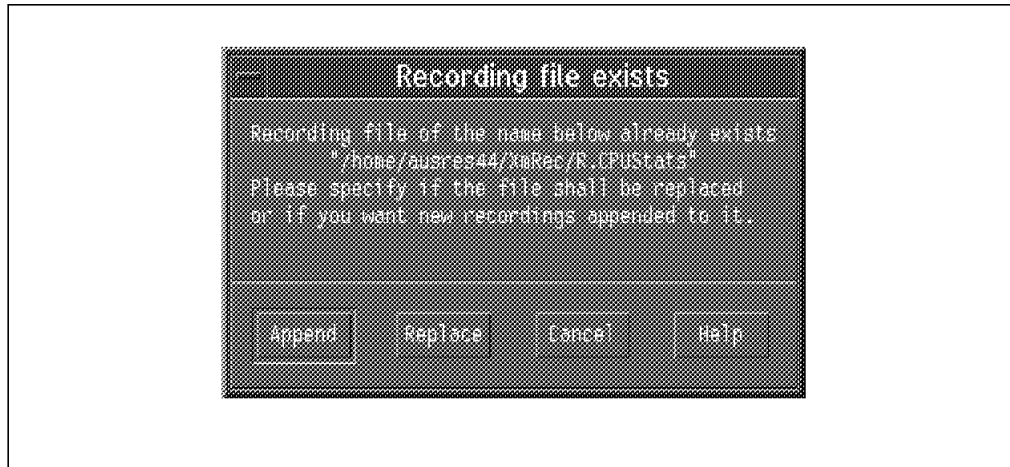


Figure 70. File Exists Pop-up Window

To remind you that recording is in progress, a symbol illustrating a tape reel is shown in the lower-right corner of all instruments (except state light instruments) when recording is active.

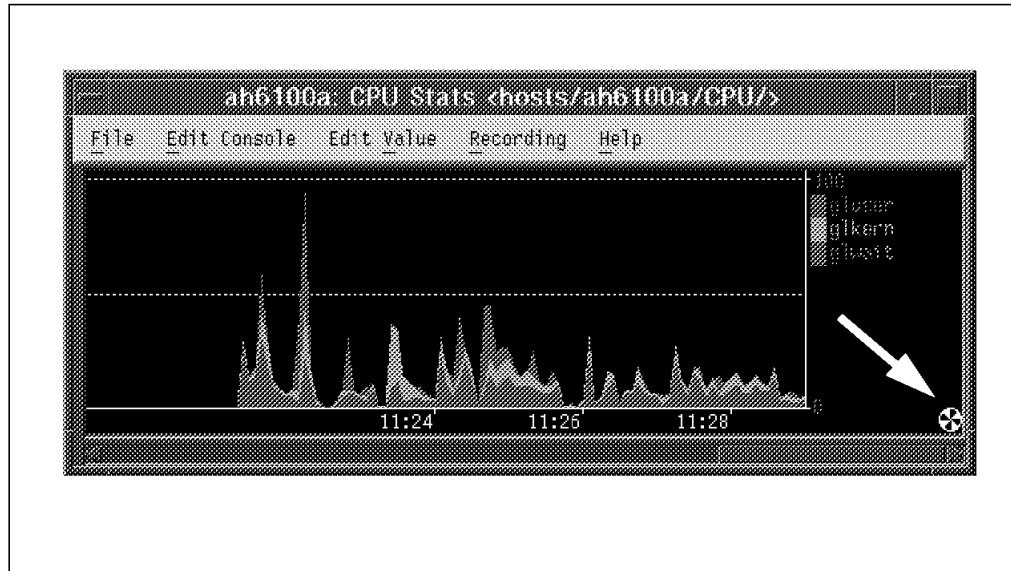


Figure 71. Console Showing Tape Reels

To stop recording, select **Recording**, then either **Console Recording** or **Instrument Recording** and then select **End Recording**.

5.1.1.1 Annotation Files

While recording is active, you may wish to add a comment when a particular event occurred. To do this select **Console Recording** or **Instrument Recording** from the Recording pull-down and then select **Annotation**. The following screen appears:

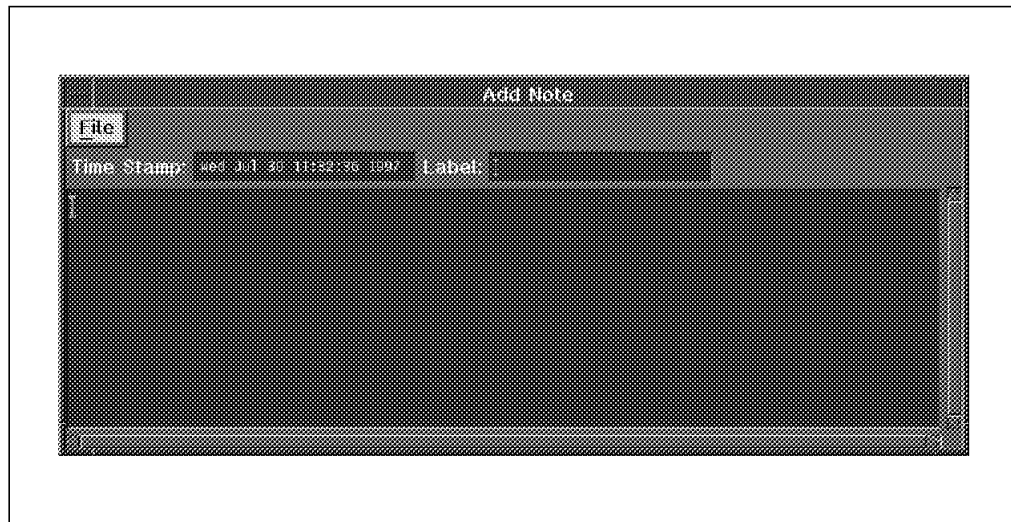


Figure 72. Adding an Annotation Window

In earlier versions, annotation files were text files that could be edited with `vi` or a similar ASCII editor. Now, annotations are special record types in recording files. They contain several structured fields and variable-length text. Because recording files are binary, special programs are required to create or delete annotation. The `xmperf`, `3dmon`, `3dplay`, and `azifo` commands all have the capabilities to work with annotations.

In the current implementation, all annotations added interactively are timestamped. Annotations have the following fields:

| | |
|------------------|---|
| Timestamp | The timestamp refers to the time in the recording when annotation was selected. |
| Status | An annotation can be either ACTIVE or marked for deletion, DELETE. This is shown in the Annotation List Window during playback. |
| Label | The abstract or label of the annotation. |
| Text | The actual text of the annotation. |

5.1.2 Recording with 3dmon

The 3dmon command may be started two different ways:

- Through the Utilities pull-down in xmpervf
- Using the 3dmon command

Any 3dmon console can be recorded in one of two ways:

1. Clicking on the **Start** icon in the bottom left of the console (1)
2. Selecting the **Begin Recording** from the **Recording** pull-down (2)

See 3.1.3, “The 3dmon.cf Configuration File” on page 54, for customizing 3dmon consoles.

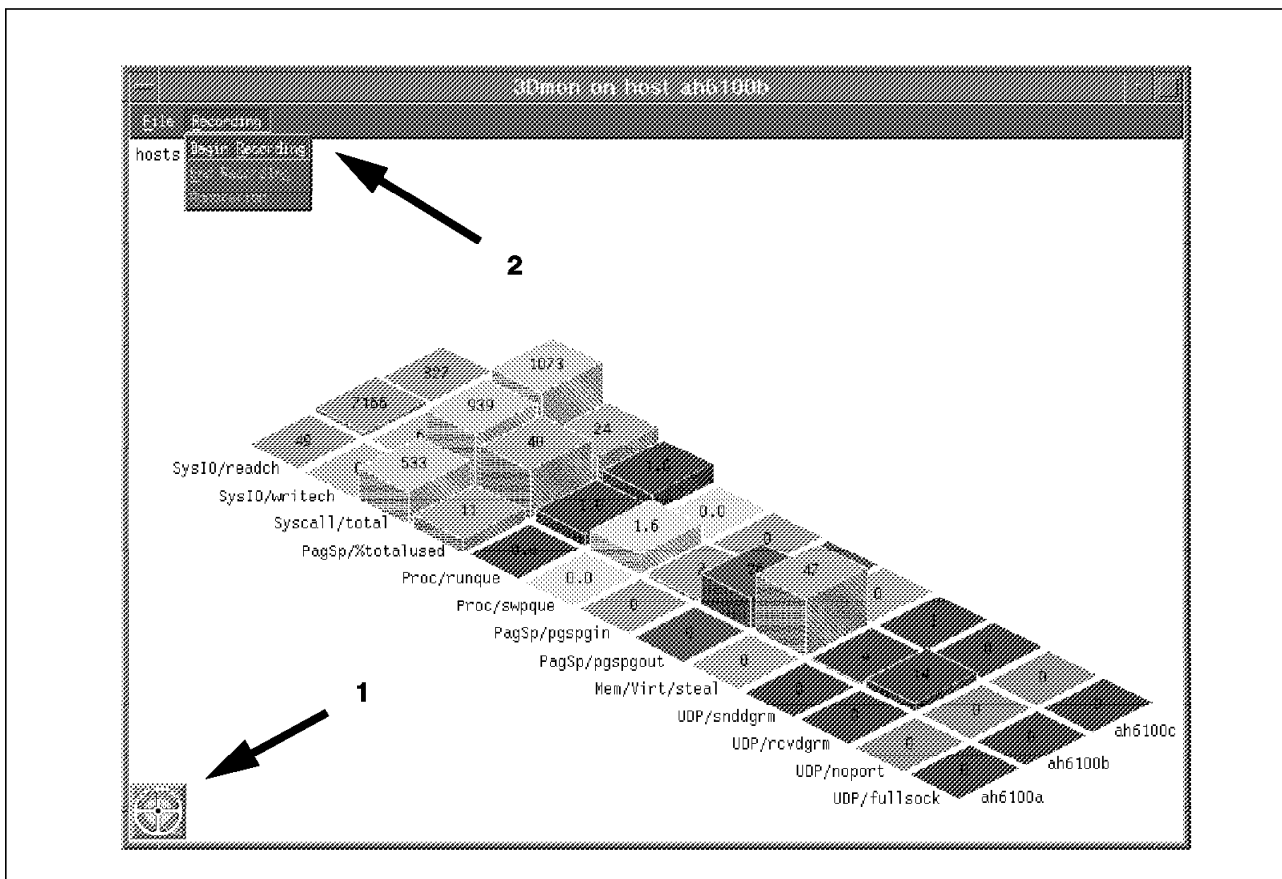


Figure 73. 3dmon Recording

When you select the **Start (1)** target icon to start recording, or select **Begin Recording** from the Recording **(2)** pull-down menu (the pull-down menu is only available in Version 2.2 or later), a pop-up window appears requesting the name of the recording file. You can accept the default name or choose a meaningful name yourself.



Figure 74. Recording Selection Window

If the recording file exists, you will be prompted whether or not you want to overwrite the file. If you select **Cancel**, you will again be presented with the pop-up window requesting the recording file name.



Figure 75. Output File Exists Window

To stop recording, you can either click on the **Stop** target icon, or select **End Recording** from the Recording pull-down menu. After the recording has been stopped, the source will be found in the specified output file.

The **Annotation** feature described above is also available from the Recording pull-down menu of 3dmon.

5.1.3 Recording with ptxrlog

The `ptxrlog` command can produce recordings in ASCII format, which allows you to print the output or post-process it, or it can produce a recording file in binary to be viewed with `azizo` or `xmperf`.

You may use the following options with this command:

- m** Manual input list of statistic name.
- f** An input file (-f filename).
- mf** Manual input list of statistic name and input file.
- h** Uses the specified as local host (-h hostname).
- i** The default recording interval is 2 seconds, but may be changed.
- b** By default, ptxrlog begins recording immediately. If a deferred recording is desired, the -b flag must be followed by hhmm, where:
 - hh** Hours in 24 hour time (midnight is 00).
 - mm** Minutes uses the specified as local host (-h hostname).
- e** By default, recording will continue for 12 hours. For shorter recording times, the -e flag must be followed by hh.mm, where:
 - hh** Number of hours to record.
 - mm** Number of minutes to record.

Refer to the *Performance Toolbox for AIX: Guide and Reference*, SC23-2625, Chapter 9 for more detailed information on all options with ptxrlog.

Below is a sample input file ptxrlog.in for ptxrlog. It contains only the list of statistics to record.

```
CPU/gluser
CPU/glkern
CPU/glwait
CPU/glidle
```

After running ptxrlog -f ptxrlog.in -i 5 -e 00.01 as the command, the following was output to the screen:

```
> ptxrlog -f ptxrlog.in -i 5 -e 00.01

Metrics Provided By Hostname: ah6100a
      CPU      CPU      CPU      CPU
Timestamp  gluser  glkern  glwait  glidle
1997/07/24 17:11:41  1.50   1.07   0.00   97.43
1997/07/24 17:11:46  2.60   1.40   0.00   96.00
1997/07/24 17:11:51  1.00   1.00   0.00   98.00
1997/07/24 17:11:56  1.60   0.40   0.00   98.00
1997/07/24 17:12:01  1.00   0.80   4.20   94.00
1997/07/24 17:12:06  1.00   0.80   0.00   98.20
1997/07/24 17:12:11  1.00   0.80   0.00   98.20
1997/07/24 17:12:16  1.40   0.20   0.00   98.40
1997/07/24 17:12:21  1.40   0.60   2.60   95.40
1997/07/24 17:12:26  1.60   1.00   0.00   97.40
1997/07/24 17:12:31  1.20   0.60   0.00   98.20
1997/07/24 17:12:36  1.20   0.40   0.00   98.40
1997/07/24 17:12:41  1.80   0.40   0.00   97.80
```

Figure 76. Output from ptxrlog

The invocation of ptxrlog can be automated by placing an entry on cron.

5.1.4 Recording with the xmsservd Daemon

The xmsservd daemon can act as a recording facility and is controlled through the xmsservd.cf configuration file. Configuration of that file is discussed in 4.1.2, “The xmsservd.cf File” on page 76.

5.1.4.1 Script to Create Custom xmsservd.cf

There are some basic performance statistics that most system administrators will want to monitor: CPU usage, memory usage, disk utilization, paging statistics, and network interface errors. Unfortunately, one configuration file may not work on another system. If the configuration file references a statistic that does not exist for that system (for instance, hdisk20), recording will not be started. For that reason, we have included a script to build a custom xmsservd.cf file for each system.

The script examines which disks and network interfaces are available, then build the configuration file. The script is called mk.xmsservd.cf. See A.1, “Sample Script to Create a Customized xmsservd.cf File” on page 173, for the complete text of the script.

```
USAGE: mk.xmsservd.cf [ -c | -f ]
mk.xmsservd.cf will build an xmsservd recording file
  -c check configuration file with xmsscheck
  -f forces defaults to be used - no prompts
      output_file=./xmsservd.cf
      retain 14 days of data
      1 day of day per file
      sample every 5 minutes
      record Mon-Fri
      record 7am-7pm
```

Figure 77. Usage Syntax for mk.xmsservd.cf

5.2 Playback

Various graphical playback methods are available depending on how the recording files were created. Some examples are:

- xmperf
- 3dplay
- azizo

5.2.1 Playback with xmperf

Playback is initiated from the File pull-down menu of the main window in xmperf. When you select the **Playback** menu item, you are presented with a list of files available for playback. The list consists of all files in the \$HOME/XmRec directory that begin with R.

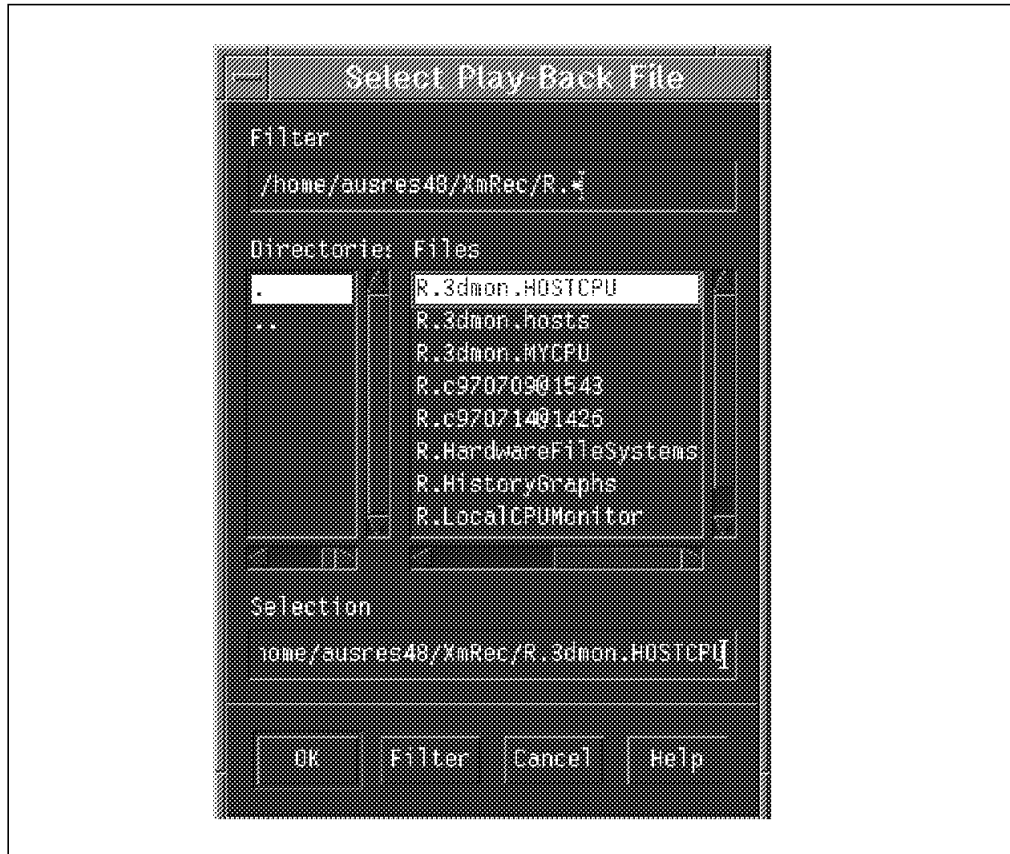


Figure 78. The xmperv Playback File Select

The xmperv command can be used to playback recording files created with any of the Performance Toolbox recording utilities, but only those created with xmperv will playback in the same style as the original recording.

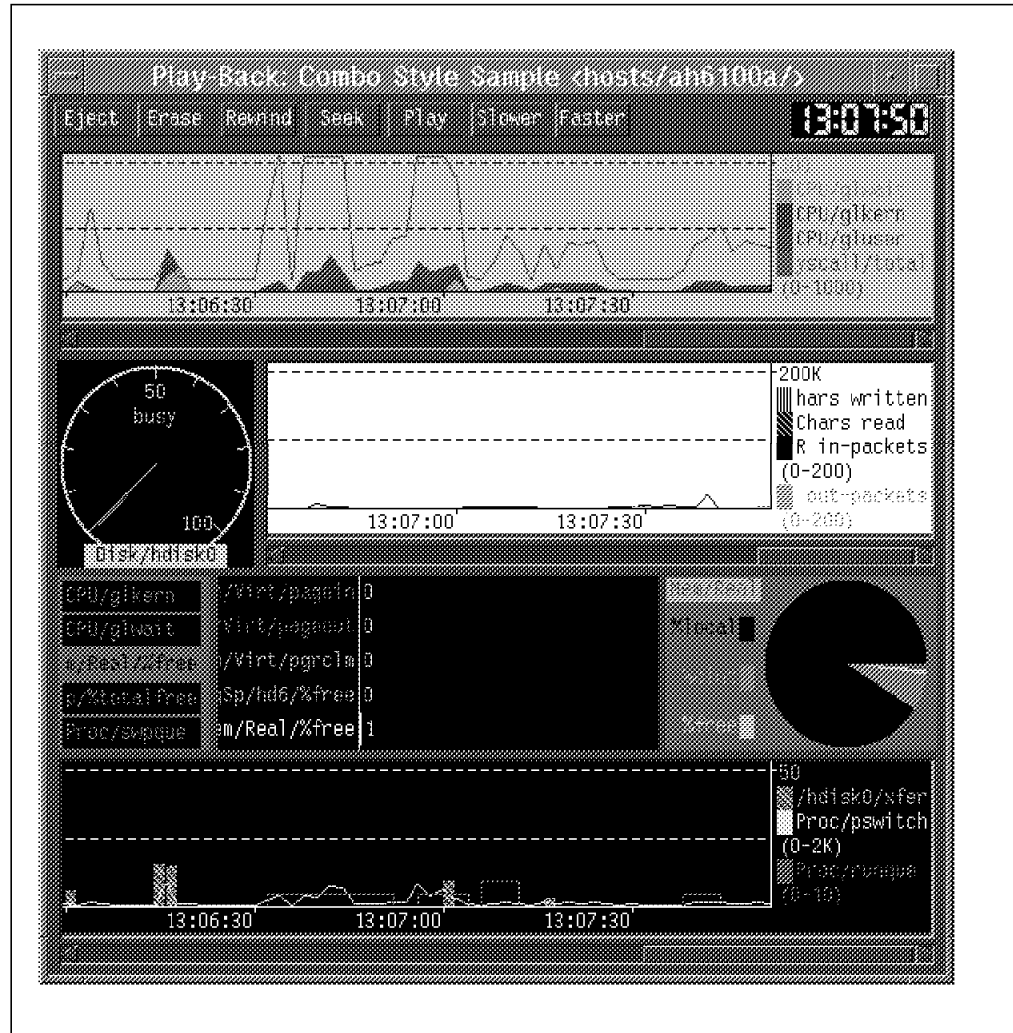


Figure 79. The xmpert Playback Window

The xmpert playback window contains a row of buttons:

- Eject** Immediately stops playback, closes the console and closes the playback file.
- Erase** Allows you to erase a playback file.



Figure 80. The xmpert Playback Erase File

- Rewind** Resets the console by clearing all instruments and rewinds the recording file to its start.
- Seek** Pops up a dialog box that allows you to specify a time you want to seek to in the playback file.

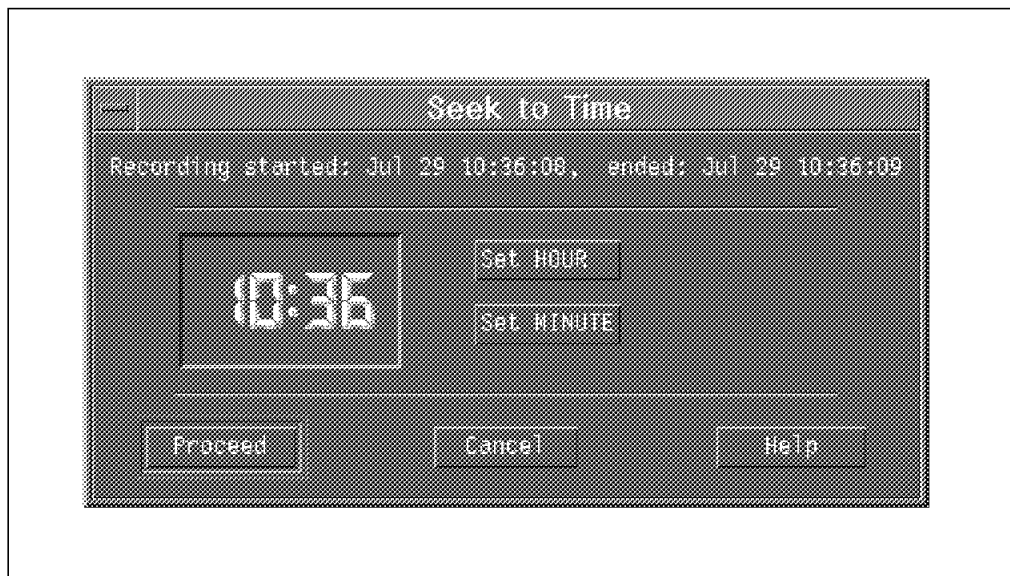


Figure 81. The xmpert Playback Seek

- Play** Starts playing from the current position in the playback file. While playing, the button's text changes to Stop to indicate that playing can be stopped by clicking the button again.
- Slower** Click on this button to cut the playback speed to half of the current speed.
- Faster** Click on this button to double the current playback speed.
- 00.00.00** At the far right is a digital clock. It shows the time corresponding to the current position in the playback file or zeroes if at the beginning of the file.

The playback pop-up menu can be displayed in two separate ways:

1. By clicking and holding down the right or left mouse button in the playback window.
2. By clicking the center mouse button once and then making the selection with another press of any mouse button.

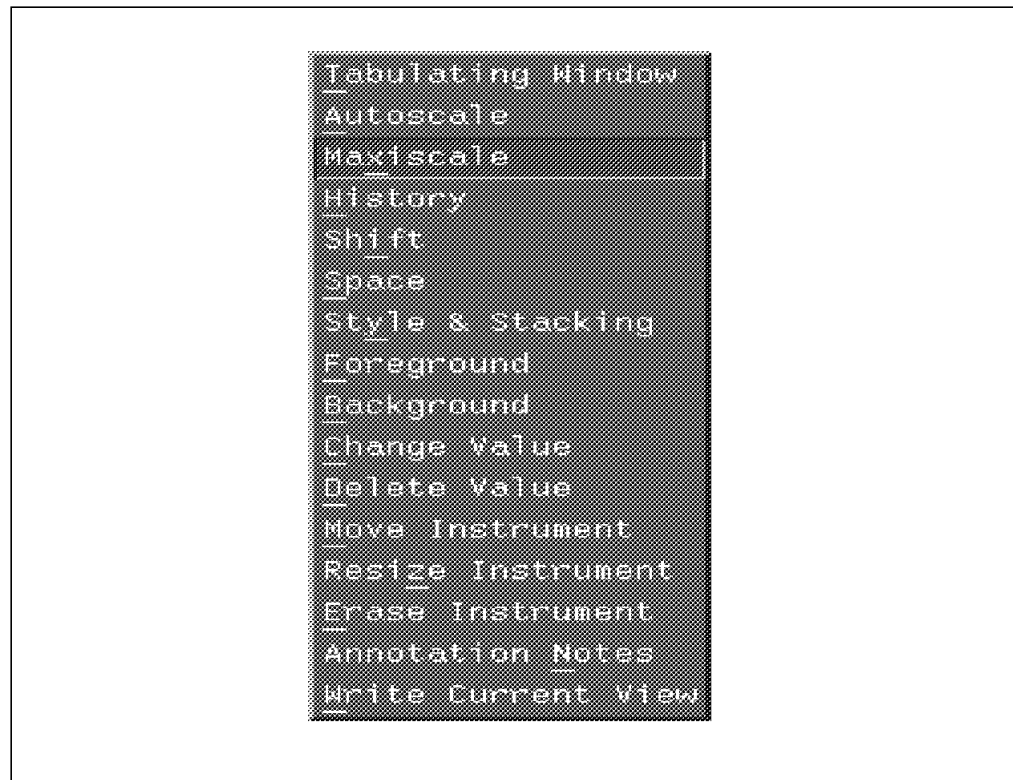


Figure 82. The xmperf Pop-up Playback Menu

The pop-up menu bar is used to modify the console or instrument to the required detail. Refer to the *Performance Toolbox for AIX: Guide and Reference*, SC23-2625, Chapter 3 for more detailed information on each specific option with this window.

5.2.2 Playback with 3dplay

The 3dplay command is used to playback recording files created with 3dmon.

The 3dplay command can be invoked any of three ways:

- | | |
|---------------------|---|
| xmperf | from the Utilities pull-down menu 3-D Play-Back (3dplay) . |
| 3dmon | from the File pull-down menu 3dmon Playback . |
| Command Line | The 3dplay command. |

Note

The 3dplay command is available with Version 2.2 and later versions.

When 3dplay is invoked without any options, a file selection window is displayed. All files in the \$HOME/XmRec directory that begin R.3dmon.* are displayed for selection.



Figure 83. 3dplay Select Recording File

Once a valid recording file is selected, the playback window is displayed.

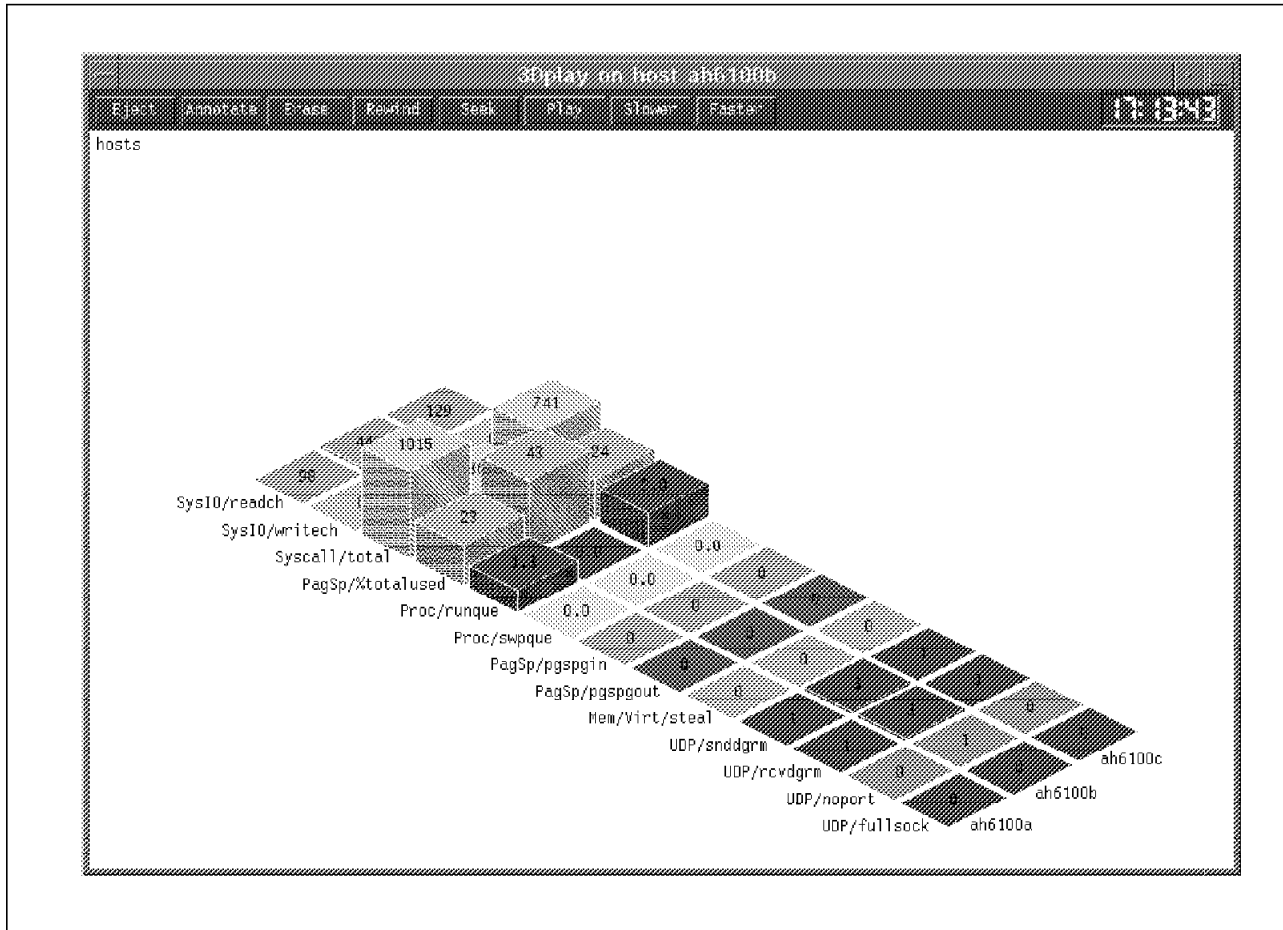


Figure 84. 3dplay Playback

The playback buttons are the same as those for xmpervf, with the addition of an Annotate button. See 5.2.1, "Playback with xmpervf" on page 99, for details.

5.2.3 Playback with azizo

The name azizo stands for Analysis Zoom In Zoom Out.

The azizo command allows you to graphically view binary recording files. Binary recording files can be created by the xmpervf or 3dmon programs during monitoring, by the xmservd daemon at any time it is running, by the a2ptx program from ASCII files that adhere to a specific format, or by the ptxrlog program.

The azizo command can be invoked in one of two ways:

xmpervf From the Utilities pull-down menu, select **Analyzing Recordings (azizo)**.

Command Line The azizo (options) command.

If invoked from xmpervf, a pop-up window appears on the screen.

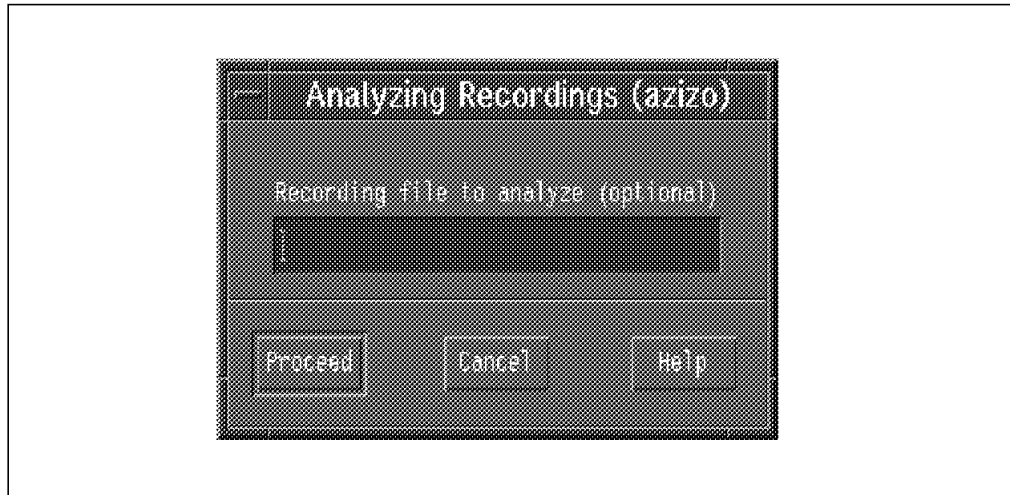


Figure 85. The azizo Recording File Pop-up Window from xmperv

You can either enter the name of a recording file to analyze and click on **Proceed**, or just click on **Proceed** and select the recording file later from the azizo screens.

The azizo main window is divided into three sections. The *Actions* section, the *Metric* section and the *Message* section. The azizo main window is shown below with the sections listed:



Figure 86. The azizo Main Window

If azizo was invoked from the command line, or you did not specify a recording file to open from the pop-up window, you will need to do so now. Click on the **Local** icon to display a file selection window.

The default filter for selecting files is `$HOME/XmRec/R.*`. This will match all recording files with their original names that were created with `xmperf` or `3dmon`. If you wish to view a recording file created from `xmservd`, change the filter to `/etc/perf/azizo.*` and click on **Filter**. The window now shows all recording files created by the daemon. If you have created a recording file with a name that does not match either of the filters, enter the recording file name in the Selection field.

Now that a recording file is open, the metric section of the main window contains one row entry for each metric found in the recording file.

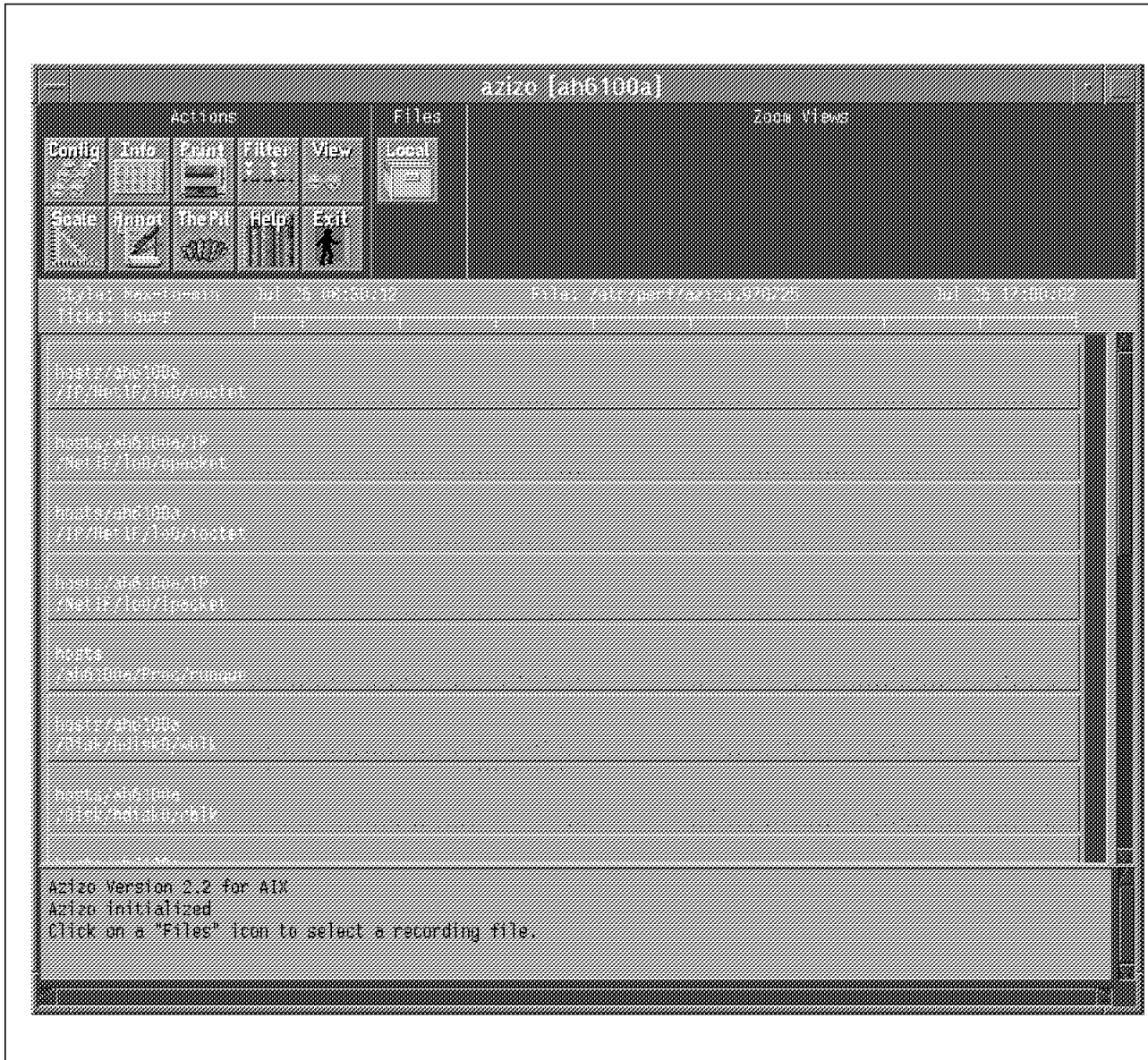


Figure 87. The azizo Main Window with Metrics

All metrics (statistics) found in the recording file are initially placed on the same azizo graph window. The graph window is divided in two sections. The left section contains all the metric names found on the graph, and the right section is the graphs on the entire time period found in the recording file.

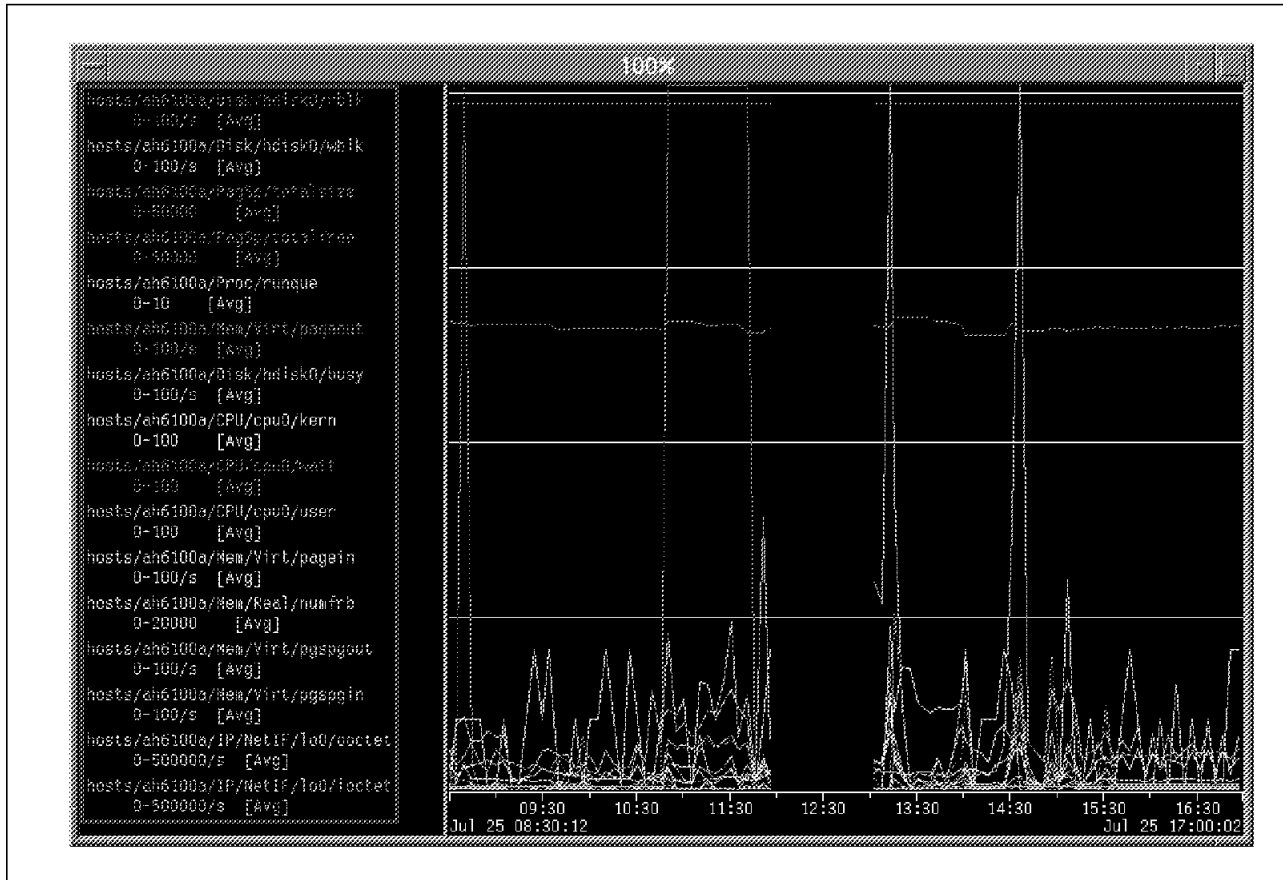


Figure 88. The azido Graph Window

The interface allows you to easily remove statistics from the graphs with a drag and drop action. Removing statistics can be done in one of two ways:

- In the main graph window, move the mouse pointer anywhere in the row displaying the metric you wish to remove. Press the middle mouse button (button 2) and drag the mouse pointer to the **Pit** icon in the Actions section. This removes the metric from both the main window and the graph window. The metric cannot be returned from the Pit.
- In the graph window, move the mouse pointer to the name on the metric in the metric list to the left of the graph. Press the middle mouse button and drag the mouse pointer to the **Pit**. This removes the metric from the graphs window, but will allow you to place it back by dragging and dropping the metric from the main window to the graph window.

You may wish to reduce the time period covered in your graph. This can be done by zooming-in on the graph window. Place the mouse pointer at the beginning of the time you want to zoom-in on. Press the left mouse button and keep it down while moving the mouse pointer to the top- and right-most period of time to include in the graph. The zoom-in dialog box now appears.

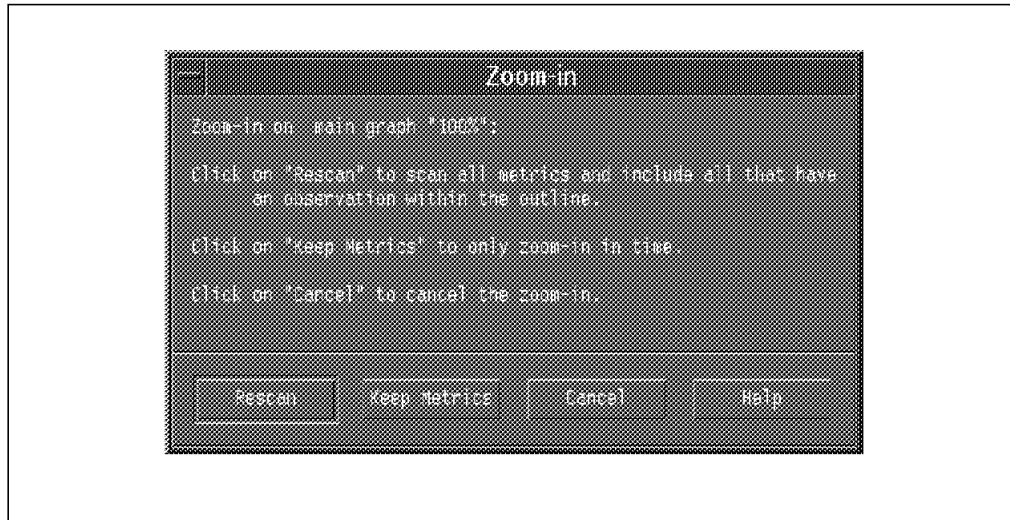


Figure 89. Zoom-in Dialog Box

Click on **Keep Metrics** to graph just the metrics in the current graph window. Selecting **Rescan** will place all metrics known in the main window on the graph.

You can keep zooming-in on the graph until you have the resolution you desire. If you want to delete a zoomed-in graph, drag it to the **Pit**.

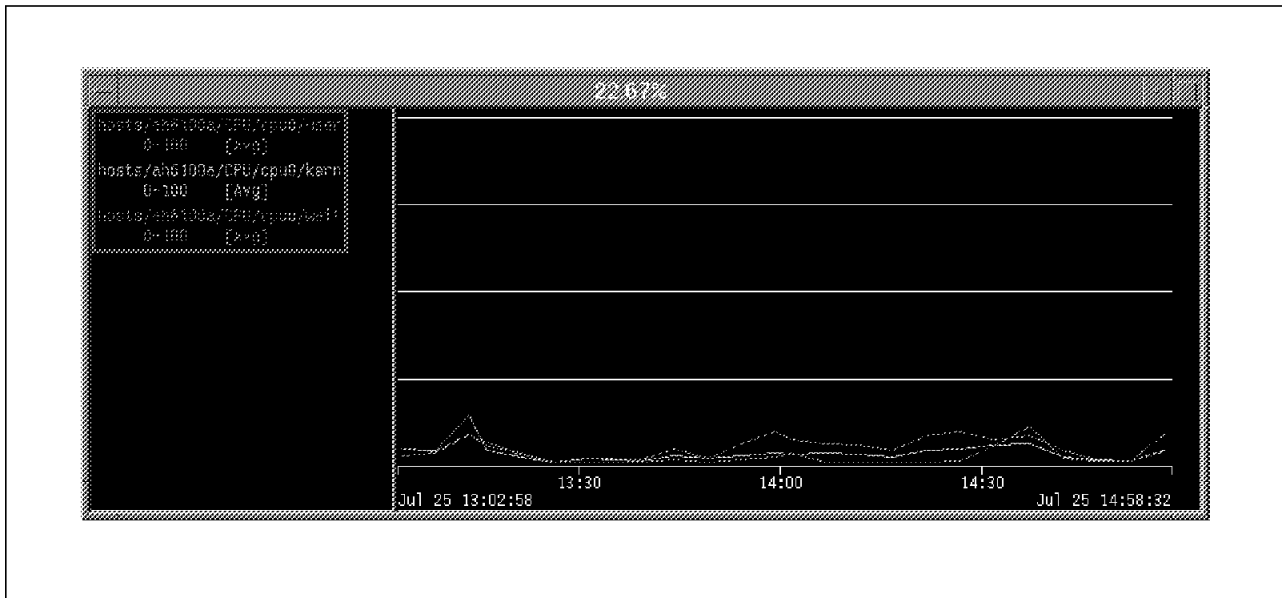


Figure 90. Zoomed-in Graph of CPU Statistics

You now have a subset of the recording file displayed on the graph. This is called *Filtering*. To store your filtered recording file, move the mouse pointer to the graph area and press and hold the middle mouse button. Then move the drag icon to the **Filter** icon in the Actions section of the main window. The filter dialog box is now displayed.



Figure 91. The azizo Filter Dialog Box

The filter dialog box has the default values for the lowest and highest time stamps to be included when the data values are copied to the filtered output file. You can change the start and end times. The time period must cover at least two seconds. The default output file name is the original recording file name with .filt appended. You can modify that field.

You can obtain statistical information for the metrics in either a main graph or zoomed in graph. Drag the graph to the **Info** icon in the Actions section of the main window. An example information window is shown below:

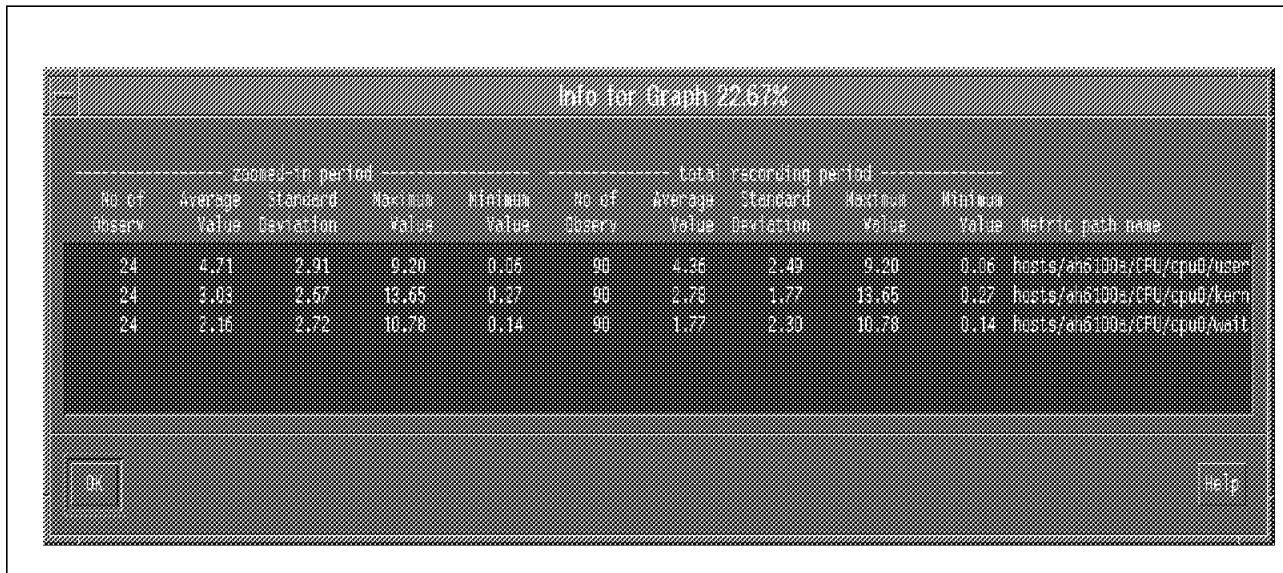


Figure 92. The azizo Info Window

Refer to the *Performance Toolbox for AIX: Guide and Reference*, SC23-2625, Chapter 10 for more detailed information on azizo.

5.2.4 Using Annotations

The most common way to view or add annotations is during playback. After selecting either **Annotation Notes** from the xperf pop-up playback menu or the **Annotate** button in the 3dplay playback window, or by dragging and dropping an azizo graph on the **Annotate** icon in the azizo main window, you see a window similar to the following:

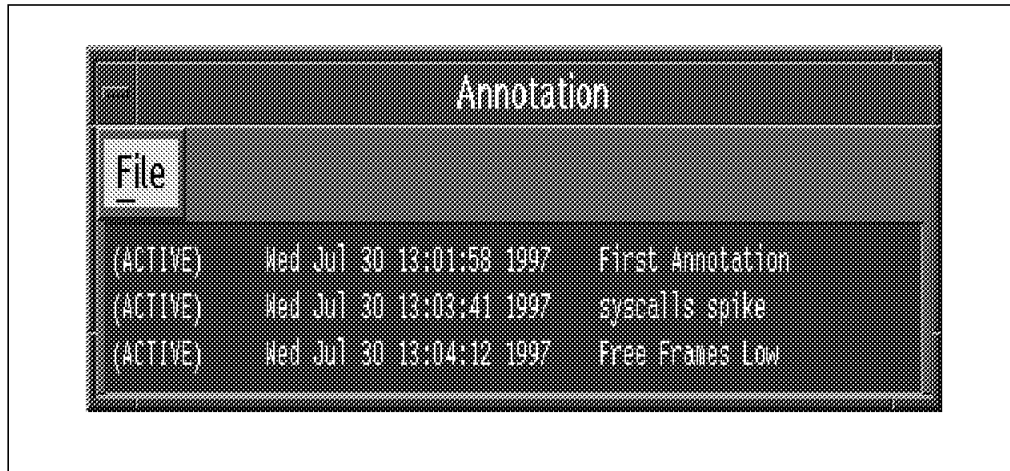


Figure 93. Annotation List Window

From the File pull-down menu, you can select:

- **Add Note**
- **Quit Annotation**

To view the annotation text, click on the annotation line, for example **syscalls spike**.

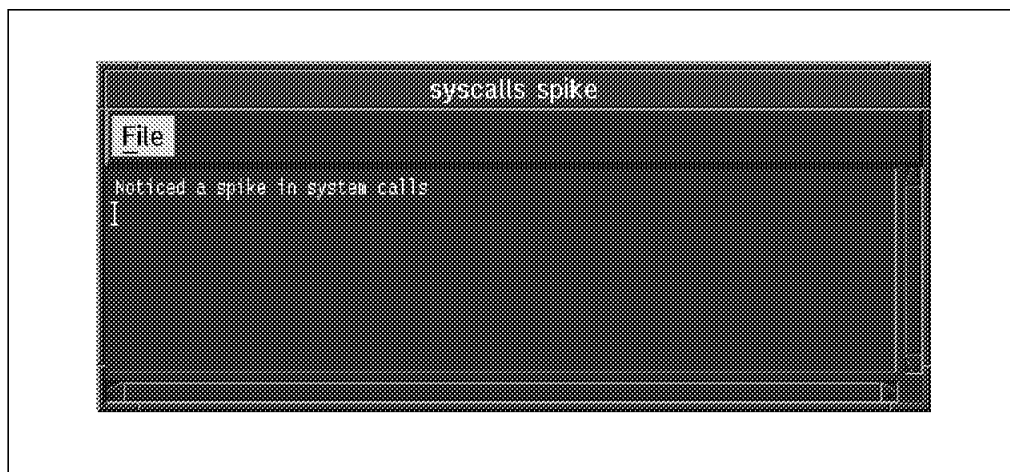


Figure 94. Annotation View Window

From the Annotation View Window, you have the following choices:

- | | |
|----------------------|---|
| Delete Note | Marks the annotation as deleted. The annotation is not actually deleted until Write Current View is selected from the playback menu. |
| Undelete Note | Removes the deletion mark. |
| Quit | Exits annotation. |

Part 2. The Performance Toolbox Parallel Extensions

Chapter 6. PTPE Overview

Performance Toolbox Parallel Extensions for AIX (PTPE) was developed to integrate performance monitoring on RS/6000 Scalable POWERparallel (SP) systems with PTX. When installed on an SP, PTPE provides specific SP performance statistics from software, such as LoadLeveler and IBM Virtual Shared Disk (VSD), and hardware (High Speed Switch) components. These statistics are available in live (current) and archived (historical) forms for use in supporting both active monitoring and comparative analysis over time.

As its name implies, PTPE extends the capabilities of Performance Toolbox (PTX) for AIX by adding monitoring *functions* and *statistics* specific to the SP. You use the same run-time instruments and consoles that PTX provides and/or use analysis, printing, and export utilities for offline processing of collected data. Performance monitoring is performed within a hierarchical framework using one central *coordinator*, one or more data *managers*, and so-called *reporters*.

PTPE provides basic command-level services for operator commands and scripts as well as an application programming interface (API) for use by AIX programs. PTPE also extends in the SP's *Perspectives* GUI providing functions to configure and control PTPE facilities.

6.1 How PTPE Works

PTPE uses a distributed approach to spread the workload and responsibilities across a number of nodes. By using this architecture, large SP complexes can be efficiently monitored without excessive burden on any one node. This distributed hierarchy has three major elements:

- A data sampler on every node configured as a *reporter*
- At least one data manager on selected node(s) configured as a *manager(s)*
- Exactly one data coordinator on the node configured as a *coordinator*

In a typical hierarchy, every node (including manager and coordinator nodes) has a data sampler daemon (reporting role). This daemon extracts performance-related information from the hardware and software components on that node based on user criteria.

At least one node in the SP has a data management role. This node manages one or more reporter nodes. This additional task includes relaying instructions to its assigned reporter nodes as well as collecting and averaging performance information from them. The manager node can also provide additional data reduction, such as, preparing summary performance figures for the nodes they manage.

The central coordinator node administers all the manager nodes and provides a single point of control for the PTPE hierarchy. This reduces the number of individual hosts that must be contacted to gather information and allows for a central process to manage all API requests. The central coordinator calculates SP-wide statistical averages of the performance summaries prepared by all the data manager nodes to provide an overall view of system activity.

This three-level hierarchy is central to the design and operation of PTPE and is illustrated in Figure 95 on page 116 below.

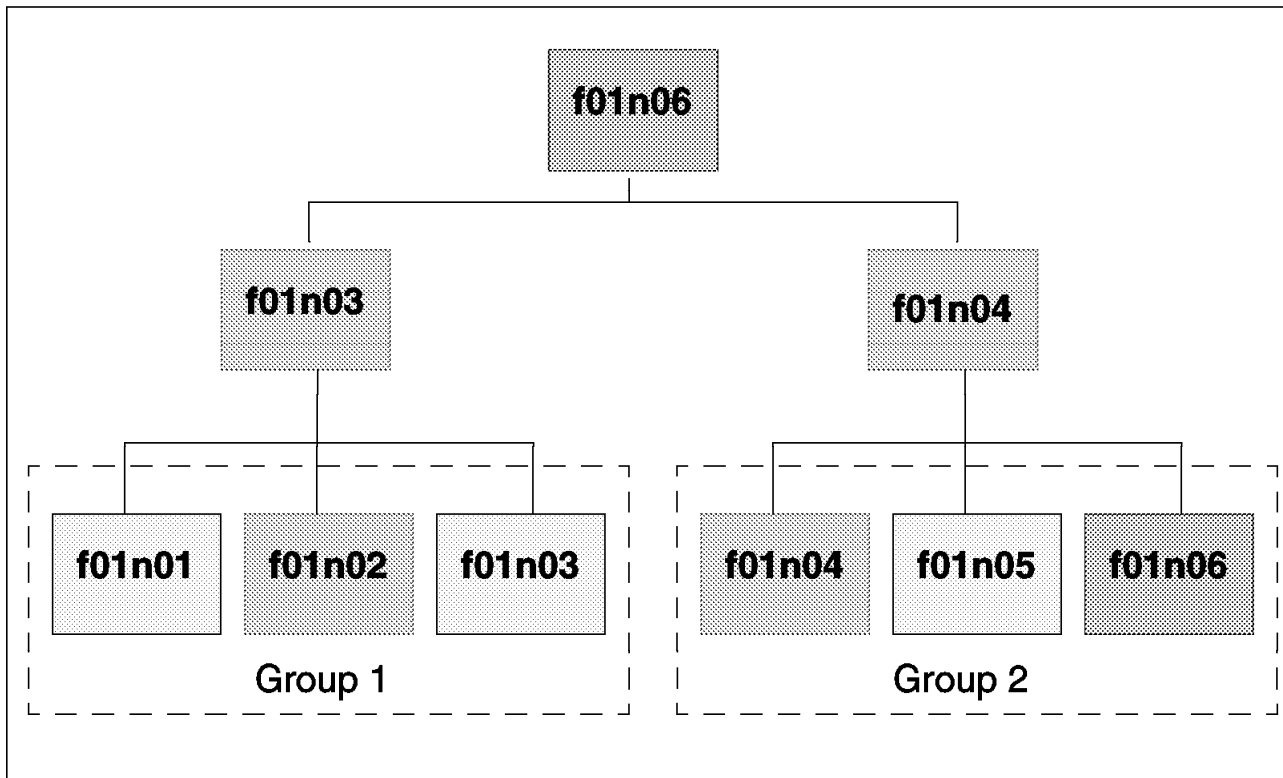


Figure 95. Sample Two-Group PTPE Hierarchy for SP Monitoring

Note that reporting groups are a logical construct used to simplify the collection and analysis of SP performance data. Reporting groups do not have to be based on frame or subnet boundaries.

For example, an SP system may have 20 nodes configured for a DB2PE database, six nodes for Lotus Domino servers, and four nodes for LoadLeveler-ISS balanced interactive access. Based on application function, three different reporting groups would be defined. The group summary reports produced by the data manager nodes would summarize application performance.

For SP environments subject to regular changes, such as online/batch schedule periods or weekly backup windows, multiple *ptpehier* input files for hierarchies and related statistical datasets can be prepared in advance and loaded as part of the application configuration.

PTPE programs may be used exclusively by so-called *ptpeuser* accounts. A *ptpeuser* is an AIX account with *perfmon* as its default group.

6.2 Using PTPE

PTPE uses a combination of configuration files and the System Data Repository (SDR) to manage configuration parameters. Each reporter node has access to a large range of local performance statistics to collect and/or archive.

By default, PTPE reporters collect data on all statistics for the node. This is an unneeded burden if you are only interested in a few specific statistics. To control the scope and detail of information collection, the reporter looks for a configuration file during startup. First, `$HOME/` is searched, followed by the `/etc/perf/` directory. You may specify a custom configuration file name or use the default, named `ptpe.cf`. If a configuration file is not found in either location, it assumes all performance metrics are to be collected. If a configuration file is located, it is read and only those statistic families listed are marked for collection.

Only specifically configured users may manage PTPE. During installation of PTPE, an admin group named `perfmon` is created if it doesn't exist already. To execute PTPE management programs, a user must have `perfmon` as their default group. It is common usage to create a specific user ID for managing PTPE.

6.2.1 Data Collection and Data Archiving

The PTPE reporter can also be configured to archive data to permanent storage. Both actions, reporting and archiving, can be configured separately for each statistic. This control can be effected at the manager level as well.

Specifying that collected data is to be archived only might be used to prevent excess LAN activity or to develop a baseline as a reference for a norm. Or performance analysis might be performed offline only on those nodes, or groups of nodes, deviating from a specified norm.

Collection and archiving settings is specified in a PTPE configuration file.

On each node, a separate logical volume should be created for archiving PTPE statistics and storing a translation table. See 7.3.3, "Insuring Adequate Log and Archive Space" on page 129.

6.2.2 Data Access and Presentation

Once PTPE has been activated and data collection started, there are a number of ways to access to the live data. Since the data format conforms to the PTX standard, any `Spmi-`(on the local node) or `Rsi-`(remote) enabled application can access the data. Typically, this might include the following:

1. An instrument console from the PTX `xmperf` tool
2. A PTX `3dmon` display with selected PTPE variables
3. A user-written application linked with the `libptpe.a` library

When using a PTX tool to display live data you can choose between the individual node statistics or the summarized versions available from the manager nodes. You can also build custom instruments for any statistics that are enabled for collection, including combinations of summary and detail data. (See Figure 96 on page 118.)

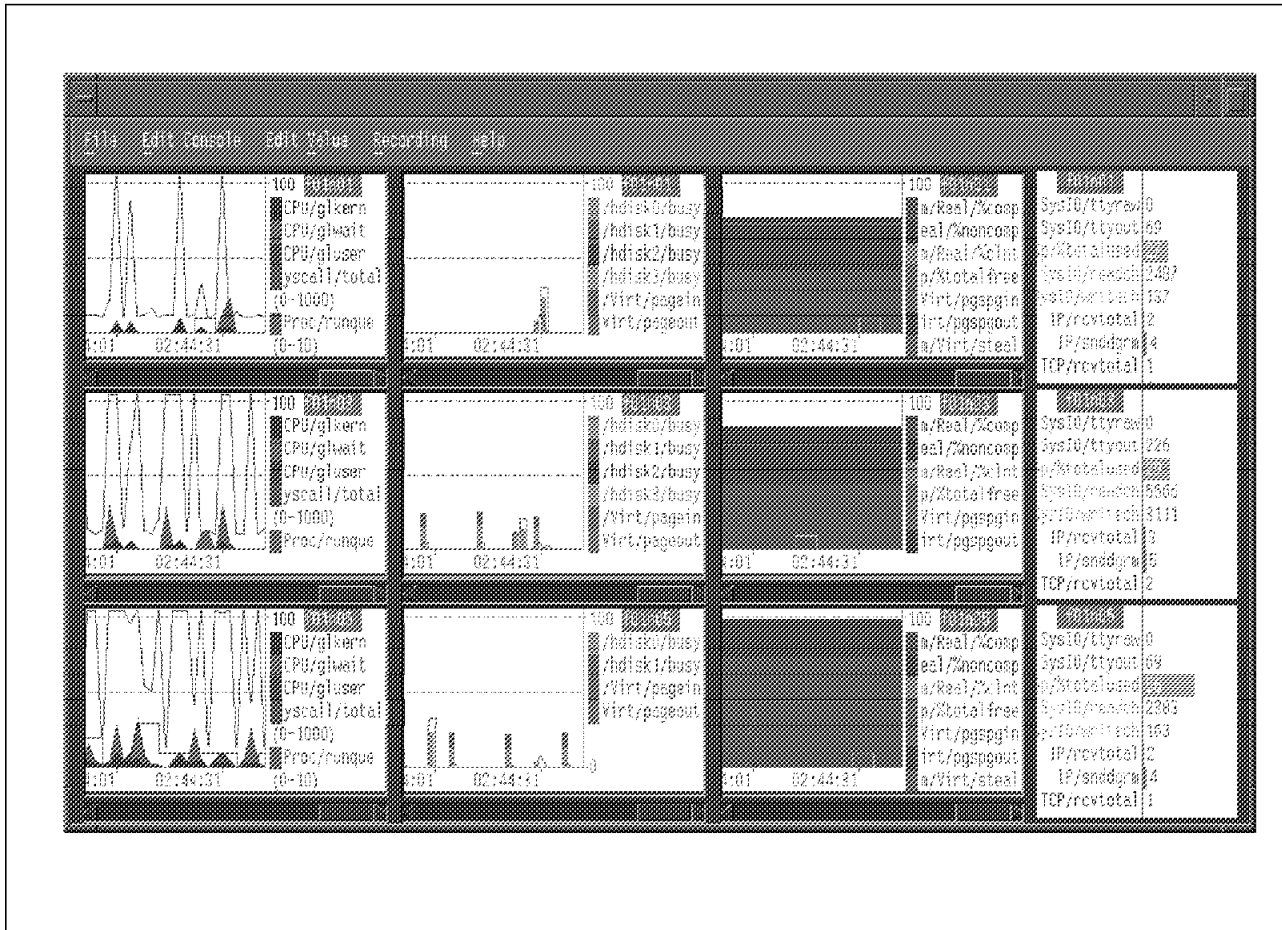


Figure 96. Multiple xperf Instruments for Nodes 1, 3, and 5

Similar capabilities are available for processing previously recorded archive data. By converting the stored data from PTP format to PTX format, you can use the PTX tools *azizo* and *xperf* to read the historical data and replay it graphically (see 5.2, “Playback” on page 99, and 8.6, “Offline Analysis of Archived Data” on page 153). In addition, you can choose to export the collected data to another program, such as a spreadsheet or statistical package, for further analysis and processing.

Finally, you can use the functions in the shared PTP library *libptpe.a* to access the archived data directly from your own application program and either display the raw data or extract and process selected data items as required.

Chapter 7. Planning, Installing and Configuring PTPE

This is an overview of the steps you perform to plan, install and configure PTPE. Further, we cover in greater detail a number of procedures you can use to install and configure PTPE on the nodes once you have determined which nodes you want to monitor. In Chapter 8, "Using PTPE on SP Systems" on page 139, the emphasis is on providing more operational examples.

7.1 Preparing Your SP for PTPE

Basic issues relevant for using PTPE for monitoring SP performance are:

- Which nodes are going to be monitored?
- Which performance monitoring role(s) will a node have (coordinator, manager, reporter)?
- How can I minimize the impact of performance monitoring on system productivity?
 - Selecting proper statistics
 - Logical/Physical hierarchy layout
- Software levels of PSSP, PTX Manager, PTX Agents and PTPE programs
 - Operating System
 - PTX
 - Perfagent.server at node
 - Perfmgr.network at PTX Manager
 - PSSP
- Distribution of management responsibility
 - Assigning roles to nodes
 - Node, CWS, or external station as PTX Manager
- Hierarchy design considerations
 - Active
 - Archive

7.1.1 Which Nodes to Monitor

Generally, unless you have a specific reason not to monitor a node, you should prepare a node for monitoring and configure a minimal set of statistics.

You might decide not to use PTPE for a node that performs a stand-alone function and you have no need of the extra SP statistics (CSS, LoadLeveler, VSD). You might be able to monitor the node with just perfagent.server functions. However, do not forget about how easy it is to configure a group of stand-alone nodes to simply archive their data locally and, consequently, relieve the LAN of any performance-monitoring load.

7.1.2 Assigning Performance Roles to Nodes

Any node that is part of PTPE will be a *reporter*. One node must be assigned as a *coordinator*, and at least one node must function as *manager*. The coordinator and manager roles are exclusive.

The manager is the focal point of data collection for a group of reporting nodes. One goal of assigning a manager is to minimize impact on network load as well as on the PTX Manager. Remember that the xmquery protocol makes use of UDP for communication. If the PTX Manager is monitoring all the nodes by itself, it may lose some statistics coming from monitored hosts.

Your goal as you plan is to identify nodes that have similar, or related, workloads and place them into groups. When the performance of the group as a whole is critical, place the manager on a node not involved in the group's production process. By keeping the manager out of the production process, the averages it computes will better reflect the group's performance. Otherwise, you must also account for possible side effects caused by the extra load of the manager.

Rule of Thumb

More reporter nodes increase data traffic and workload for a PTPE manager, while fewer nodes in a group decrease traffic and workload for a PTPE manager. Having more managers also provides greater flexibility in setting up statsets to be reported by the nodes (more groups --> more different statsets possible).

Having more PTPE managers may mean more SP nodes are needed. Further, more PTPE managers may increase data traffic between the PTX Manager and/or the PTPE coordinator.

7.1.3 Choosing PTPE Statistics

The statistics you choose determine the effectiveness of your monitoring. The statistics that need to be monitored should have some predictive value about the (pending) exhaustion of a processing resource, or, in other words, the formation of a process bottleneck. So, initial statistics always include CPU, Paging (File versus Computational), mbuf allocation, local I/O, and network adapter statistics.

Choosing the correct statistics requires an understanding of the architecture and dynamics of the information system being monitored. This should give an initial set of must-have statistics. As more experience and knowledge is gained about the information system, modifications can be made and/or alternate statistics sets might be developed. Developing this increased insight might be accomplished by reporting immediately a subset of well-recognized statistics (CPU, Paging, I/O wait, IP activity) while also locally archiving additional statistics on the individual nodes and/or the reporter. Offline analysis of these additional statistics can be used to develop better thresholds, or predictors, of pending resource exhaustion.

Part of the planning is identifying nodes that are regularly being reloaded with a new application, or switching from batch to interactive, and so forth. These nodes require more planning about how and when to consolidate reports and summaries.

Remember, the statistics you choose to monitor will have the greatest impact on the effectiveness of performance monitoring. Too few, or choosing the wrong ones, leads to missed predictors of pending problems. Collecting too many statistics may degrade the performance of the information system being monitored and require unnecessary space to collect and/or archive statistics.

7.1.4 Choosing a PTX Manager

In recent months, the PTX agent (2.1.5 and 2.1.6) for AIX 4.1 has been brought up to the functionality of Release 2.2 of the PTX agent for AIX 4.2. If you are going to manage from a PTX 2.2.1 level manager, then we recommend that you use the `perfmgr.network.2.2.1.1` fileset as manager. For AIX 4.1.X systems, you only need to upgrade to 4.1.5 on the SP nodes to work with PTPE. Nodes managed outside the SP may remain at a lower level and still be manageable.

Table 1. AIX and PTX Levels

| Manager | | Agents | |
|--------------|------------------------------|--------------|-------------------------------------|
| AIX OS Level | <code>perfmgr.network</code> | AIX OS Level | <code>perfagent.server</code> Level |
| 4.2.1 | 2.2.1.1 | 4.1.4 | 2.1.4.5 ^a |
| | | 4.1.5 | 2.1.5.0 ^b |
| | | 4.1.5 | 2.1.6.0 |
| | | 4.1.5 | 2.1.6.1 |
| | | 4.2.0 | 2.2.0.3 |
| | | 4.2.0 | 2.2.1.2 ^c |

a. The 2.1.4.x level range is between 2.1.4.4 and 2.1.4.10.

b. Level 2.1.5.0 requires AIX 4.1.5.

c. Once you start with `perfagent 2.2.1.X`, you must have at least 2.2.1.2 for PTPE to function.

7.1.5 Overview of Requisite Software

The filesets needed are:

| | |
|--------------------------------------|---|
| <code>perfmgr.*</code> | Installed on PTX Manager |
| <code>perfagent.server</code> | Installed on all systems being monitored |
| <code>ptpe.program</code> | Installed on all nodes and the Control Workstation (CWS) |
| <code>ptpe.gui</code> | Installed on CWS and any nodes where Perspectives is installed or likely to be executed |
| <code>ptpe.doc</code> | Installed on documentation system, maybe on the CWS |

7.1.5.1 Performance Manager Host

We tested only with a performance manager at levels 2.2.1.0 and 2.2.1.1. The manager at level 2.2.1.0 doesn't work with several `perfagent.server` releases. This is caused by a binary compatibility problem that was fixed in the fileset `perfmgr.common.2.2.2.1`.

Note that the host chosen as manager does not have to be part of the SP environment, but may be a separate workstation. We used model 43Ps extensively for this. Working at a great distance (over a WAN) is also possible.

We also worked on an SP in the Netherlands by exporting the DISPLAY to a local 43P.

7.1.5.2 Performance Agent Filesets

As you can see in Table 1 on page 121, we tested with a great variety of perfagent.server releases. The only problem found was with the perfagent.server.2.2.1.0. This fileset version is missing an external symbol in the libSpmi.a library so that PTPE daemons cannot be loaded to run.

How do you determine if PTPE is failing because of a bad libSpmi.a?: The way to do it is to run this command on a node:

```
/usr/lpp/ptpe/bin/spdmspld 0 0
```

The syntax doesn't really matter. If the loader immediately reports a problem, then you know you have a bad library. If it doesn't, kill the command, and consult the problem determination information in the PTPE manual (you now have a somewhat more complex problem to debug).

The level of perfagent.server recommended for AIX 4.1.X is perfagent.server.2.1.6.1 and for AIX 4.2.X either perfagent.server.2.2.0.3 or perfagent.server.2.2.1.2 or higher.

7.1.5.3 PTPE

The PTPE LPP consists of a number of filesets:

- ptpe.program
- ptpe.gui
- ptpe.docs

The ptpe.program fileset is installed on the Control Workstation and all nodes that participate in PTPE monitoring. The ptpe.gui fileset is the addition to Perspectives GUI that may be used to manage PTPE functions. The ptpe.docs fileset is the online documentation.

We worked with three different levels of ptpe.program: ptpe.program.1.1.0.0, ptpe.program1.1.0.2 and ptpe.program1.1.0.3. These three levels are not dependent on the level of perfagent.server but on the level of the ssp.* filesets installed on the CWS. In other words, the level of ptpe.program you use is determined by your PSSP software Program Temporary Fix (PTF) level.

7.1.5.4 Locating the Latest PTF Levels

One service of IBM we used to control our software levels is at URL <http://service.software.ibm.com/aix/launch.html>. Starting at this page, you can choose your country. This new page is the page to add as a bookmark in your browser. Anyone may download fix_dist and/or the electronic fix distribution. If your country is not the United States, you may want to look at <http://service.software.ibm.com/aix.us/support?lang=english> for the other links to downloadable software. We didn't check all the country pages for their links.

7.2 Planning PTPE

To help planning, we developed a couple of tables you can copy or use as a model for a spreadsheet.

Table 2 on page 124 helps plan and register PTPE hierarchies. One column in both tables is labeled *Primary Task*. With the exception of the PTX Manager, we assume that every host or node has a function more important to the information services provided than that of performance monitoring. Rather than organize a hierarchy according to a node's physical location in a frame, or in an IP (sub)net, you may see a valid organization based on a node's primary task.

Table 2. Sample Table for Planning PTPH Hierarchy

| Frame | Node | Primary Task | Performance Role | AIX Level | perfagent level | Comments |
|-------|------|--------------|------------------|-----------|-----------------|----------|
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

Table 3 can be used to record the various software levels of the various performance management and/or software installation functions that are needed to effectively manage PTPH installation, configuration and operation.

Table 3. Sample Table for Planning Managers, Servers, and Coordinators

| System | Primary Task | Performance Role | AIX Level | perfagent level | Comments |
|--------|--------------|------------------|-----------|-----------------|------------------------------------|
| | | PTX Manager | | | perfmggr.network is at level _____ |
| | SP CWS | | | | |
| | boot/server | | | | |
| | | PTPE coordinator | | | |
| | | PTPE manager | | | |
| | | PTPE manager | | | |

7.2.1 Planning a Hierarchy

We have a some suggestions for developing your initial hierarchies.

Determine the role(s) a node should fulfill.

1. First decide which node will coordinate.

The coordinator node needs to be reliable and should have sufficient disk space for any summaries and reports kept within PTPE. Remember, all managers are reporting to it. It may be the only node a PTX Manager routinely monitors.

2. All nodes are reporters.

Our advice is to make all node reporters unless you have a specific need to exclude a node. If you do exclude a node, consider whether you are excluding it from installation as well as data collection and archiving. If you exclude installing PTPE, do you also want to remove perfagent.server as well? Our advice is to install PTPE on the node and leave perfagent.server installed. Should the need ever arise to include/exclude a node from PTPE monitoring, any ptpeuser can do this without any special rights and/or without possible disruption of the node's primary task.

3. Which nodes will be managers?

Remember that managing other nodes require CPU, disk and network resources. A manager node does not need to be within the same frame as the nodes it manages, but keeping a manager within the same frame as the reporters should make it possible to minimize the impact on network resources otherwise needed for the production processes (as opposed to the overhead of system management processes such as PTPE).

4. Which system/host acts as PTX Manager?

The PTX Manager is not necessarily a node within the SP. More importantly, the PTX Manager should have good connectivity to the PTPE coordinator and PTPE managers. In principle, we tried to keep the manager outside the SP (see Figure 101 on page 139).

In Figure 101 on page 139, we show a hierarchy with a single manager, and in Figure 95 on page 116, we show a hierarchy with two managers.

7.2.1.1 Hierarchy Guidelines

Another part of the job to clean up the script is performed by adding some checks before creating a new logical volume. If the logical volume already exists, a number of error messages are reported. These do not cause the script to fail. They just aren't very pretty.

In the following, the numbered paragraph is followed by some premises that may be used collectively or separately as a hierarchy guideline.

1. Group nodes by some user criteria into reporting groups.
 - Nodes do not have to be within the same frame.
 - Reporting groups are not necessarily related to nodegroups.
 - Groups need to be within the same partition.
2. One node in each reporting group is selected as the data manager.
 - Nodes outside of a reporting group cannot be the data manager.

- All data managers must also be reporting nodes within their group.
3. One node is selected as the central coordinator.
 - This must be a node within the hierarchy.
 - This node cannot also be a data manager.

7.2.1.2 Performance Considerations

The additional workload and network traffic increase due to PTPE activity should be a factor in determining the structure and assignment of your reporting hierarchy. The increase depends on the range of statistics collected by reporter nodes, the number of reporter nodes assigned to each data manager node, and the number of data manager nodes the central coordinator has to oversee. The following general guidelines can be offered:

1. Limit manager nodes to between 4-24 reporter nodes each, depending on manager node type and average workload.
2. Try to keep group traffic within the same network (segment/ring) to avoid overloading intermediate bridges and/or routers.

As much as possible, don't let measuring performance have an (adverse) masking effect on the true system baseline.

7.2.2 Preparing the Installation

Before you actually begin installing PTPE on nodes, you need to have the `perfagent.server` and `ptpe.program` filesets available on a Network Installation Manager (NIM) software server. We tried several methods of installing PTPE, but using NIM gives the greatest flexibility and ease of management, especially when a newer release PTF level comes out.

Table 4 shows how we prepared the CWS as a NIM master server to distribute PTPE to the nodes. Don't install all the software images just yet. Read 7.3.1, "Installing the Filesets" on page 127 first.

Table 4. Table of NIM Resources Defined to Ease Installation of PTP

| FileSet | Location | Server | NIM Resource |
|-------------------------------------|---|--------|------------------------------|
| <code>ptpe.program</code> | <code>/spdata/sys1/install/lpp_PTPE</code> | master | <code>lppsource_ptpe</code> |
| <code>perfagent.server.2.1.X</code> | <code>/spdata/sys1/install/lpp_ptx21</code> | master | <code>lppsource_ptx21</code> |
| <code>perfagent.server.2.2.X</code> | <code>/spdata/sys1/install/lpp_ptx22</code> | master | <code>lppsource_ptx22</code> |

7.3 Installing PTPE

Prior to installing PTPE on a node, PTPE must be installed on the Control Workstation. There are several additional requirements needed for using PTPE that are taken care of automatically if you install PTPE successfully on the CWS first. These include assigning an IP port that is used by PTPE to communicate between the different nodes and storing this in the SDR. Further, the root account is set up as the initial `ptpeuser` account by creating the `perfmon` group and adding `perfmon` to root's group membership.

Installing PTPE in an SP environment is a two-step process. First, PTPE is installed on the Control Workstation to insure integration with PSSP and the SDR entries. Initially, we performed PTPE configuration on the CWS and set the initial hierarchy in the SDR.

7.3.1 Installing the Filesets

Although PTPE may be installed directly from the installation device (generally a tape or CD-ROM), we recommend that you copy the installation image to the `pssp/lppsource` directory before installing it on the Control Workstation. After you have done this, you can make your NIM `lpp_source` directories. We made them in the `/spdata/sys1/install` directory, but you can make them anywhere. We found it convenient to be in the same filesystem (`/spdata`) so that we could work with hard links.

Note

You don't need to install the base `perfagent.server` filesets. These are already included as part of the `lpp_source` to support the SP nodes.

Having a separate directory for the PTX agents makes it simpler to maintain PTF levels. It is not necessary to customize the node's spots (assuming, in NIM's terminology, that all the SP nodes are stand-alone machines). All that is required is to run the program `inutoc` for each of the directories.

After these filesets are installed, you can define the NIM resources. The command to define a NIM resource is simple:

```
nim -o define -t <type> [-a <attribute_id>=<attribute_value>] <resource>
```

To define a NIM resource of type `lpp_source`, the following command is necessary:

```
nim -o define -t lpp_source -a server=<nimserver> -a location=<path> <resource>
```

We placed links to `ptpe.program` and its PTFs in a separate directory (`/spdata/sys1/install/lpp_PTPE`) and links to `perfagent.server` and their respective PTFs in the directories `/spdata/sys1/install/lppagent21` and `/spdata/sys1/install/lppagent22`, respectively

To define these resources in NIM, we used the following commands:

```
# nim -o define -t lpp_source
  -a server=master \
  -a location=/spdata/sys1/install/lppagent21 lppsource_ptx21
# nim -o define -t lpp_source \
  -a server=master \
  -a location=/spdata/sys1/install/lppagent22 lppsource_ptx22
# nim -o define -t lpp_source \
  -a server=master \
  -a location=/spdata/sys1/install/lpp_PTPE lppsource_ptpe
```

7.3.2 Planning for PTPE Users

What you need to plan for is whether this is to be an account that is created on the Control Workstation and distributed to nodes through the SP file supper subsystem or if the account(s) is/are to be created separately on an individual node or nodes.

7.3.2.1 PTPE User Account Creation

Creating an additional *ptpeuser* account is not difficult, but due to PSSP requirements, two criteria must be met when establishing the account.

1. The user must have the perfmon group ID (GID) in their group list.
2. The effective user ID (UID) must be 0 (zero).

The perfmon group is required in order to restrict access to PTPE resources and commands to authorized users only. The UID of 0 is required because some PTPE commands need to write data to the SDR, and at this time, only root users are allowed write access to SDR classes.

The easiest way to meet these criteria is to build the additional *ptpeuser* account using the normal AIX methods (*smi*t or command line) and insure that the perfmon group is specified. Then, as root, edit the */etc/passwd* file and change the assigned UID for the account to 0. Then propagate (if necessary) the changes to the required nodes.

We prefer setting up a separate *ptpeuser* account on at least one node (for example, the coordinator node) so that PTPE may be managed without the unnecessary usage of root access rights, and/or user access, to the Control Workstation. This is not the method discussed in the standard documentation. The standard documentation suggests the strategy of making an alias root account by editing the */etc/passwd* file and assigning the UID of 0 to the account.

```
root:!:0:0:Root account:/:bin/ksh
ptpeuser:0:11:PTPEUSER Account:/:bin/ksh
```

Note that the group ID on your system may differ with ours (for instance, perfmon on our system has GID 11).

Remember that, by default, root is set up as a *ptpeuser* account on every node *ptpe.program* is installed on, but perfmon is not root's default group.

If you prefer to limit root access, you can Set-user-ID-on-execution (SUID) for the programs as follows:

```
chown root:perfmon /usr/lpp/ptpe/bin/ptpectrl /usr/lpp/ptpe/bin/ptpehier
chmod u+s /usr/lpp/ptpe/bin/ptpectrl /usr/lpp/ptpe/bin/ptpehier
chmod o-rwx /usr/lpp/ptpe/bin/ptpectrl /usr/lpp/ptpe/bin/ptpehier
```

7.3.2.2 PTPE User Policies

Make sure there are some policies about when and how PTPE is to be managed by *ptpeuser* accounts. Note that as long as the PTPE GUI is active on any node in the SP, access to the PTPE entries in the SDR is not allowed to the command line interface and/or another PTPE GUI process.

7.3.3 Insuring Adequate Log and Archive Space

The PTPE performance data archives are maintained in the `/var/adm/ptpe` subdirectory. Depending on your configuration and usage, the amount of space required could grow quite large and consume most of the `/var` filesystem. While the PTPE is intelligent enough not to consume all the space in `/var` (it will stop archiving at the 95 percent full level), it is recommended that a separate filesystem be used to contain the performance data archive.

This can be accomplished by defining a logical volume of sufficient space to contain approximately twice (2x) the size of the largest performance data archive expected to exist on the node. This allows for later use of `ptpedump`, `spdm_dump`, and other programs that need to extract data from the archive and build an ASCII version in the same location. Then create a new JFS using this logical volume and mount it over the original `/var/adm/ptpe`. By using this technique, you can avoid impacting the other users of `/var` and easily modify the amount of space allocated to PTPE.

A good starting point would be a 32 MB logical volume. A general rule of thumb might be:

$$\text{SIZE} = 2 * (13000022 + (N * 20 * (L / I))) \text{ bytes}$$

L = expected_life_of_archive_in_seconds.

N = expected_number_of_stats_recorded.

I = recording_interval_in_seconds (see `ptpectrl -f` command).

The number 13000022 is the number of bytes reserved by the translation table itself (about 13 MB).

The number 20 is the number of bytes PTPE records for each statistic each time it records it in the archive.

Now all you need to do is to convert SIZE, which is in bytes, into blocks since SMIT uses blocks and not bytes on its logical volume panels.

7.3.4 AIX Syslog and Error Log

PTPE makes use of several techniques to notify administrators of events or unusual conditions during operation. In order to make the best use of PTPE, you must know where and how to read and interpret its messages. Like all well-behaved AIX applications, it makes use of both the AIX system logger and the error logger. By configuring your `syslogd` parameters, you can control how the loggers get routed.

```
f01n07:/u/ptpeuser tail /etc/syslog.conf
..
user.warn /tmp/db2.syslog
*.emerg;*.alert;*.err;user.none /tmp/aix.syslog
```

This example shows how the emergency, alert, and error message classes are segregated from the user database application. They are both logged to a specific file, but could also be routed to the console or sent through mail. Using entries such as these allows for PTPE messages to be located quickly should a problem arise. For example, if PTPE fails to start on a node, the AIX syslog entries will show something like this.

```
f01n07:/u/ptpeuser > tail /tmp/aix.syslog
...
Sep 05 21:15:30 f01n07 perfmon[44962]: LPP=PTPE,Fn=debug.c,SID=I,L#=207,
  spdmcold : 2516-706 Failed to read reply from f01n03 in routine
  (read_sampling_data_from_one_node()).
Sep 05 21:15:30 f01n07 perfmon[44962]: LPP=PTPE,Fn=debug.c,SID=I,L#=207,
  spdmcold : 2516-660 Socket 3 no longer connected. Peer f01n03 may have
  closed connection prematurely or died unexpectedly.
```

Another useful place to look when things don't seem to be behaving properly is in the AIX error log. PTPE uses the standard error log template to record errors and information messages using the AIX format. This means you can distinguish between informational messages and real errors, permanent and temporary errors, and errors caused by specific components. Be careful though; not all messages here are indications of an error event.

7.3.5 SDR Entries for PTPE

PTPE should always be installed on the Control Workstation first.

7.3.5.1 IP Port Definition

Part of the installation process on the Control Workstation is to reserve an IP port address that will be used for communication between the various components. This port number is placed into the SDR for reference by the nodes as PTPE gets installed on them.

7.3.5.2 Reporting Hierarchy

The current ptpehier monitoring hierarchy is stored in the SDR for reference by the nodes.

7.3.6 Pushing PTPE to Nodes through NIM

We tried various methods for installing PTPE on the nodes. Using Network Installation Manager (NIM) provided the greatest flexibility and ease of installation. We defined a number of lpp_sources to be able to install the components easily.

The command to define a NIM resource is simple:

```
nim -o define -t <type> [-a <attribute_id>=<attribute_value>] <resource>
```

To define a NIM resource of type lpp_source, the following command is necessary:

```
nim -o define -t lpp_source -a master=<nimserver> -a location=<path> <resource>
```

We placed links to ptpe.program and its PTFs in a separate directory (/spdata/sys1/install/lppPTPE) and links to peragent.server and their respective PTFs in the directories /spdata/sys1/install/lppagent21 and /spdata/sys1/install/lppagent22.

To define these resources in NIM, we used the following commands:

```
# nim -o define -t lpp_source
  -a master=server \
  -a location=/spdata/sys1/install/lppagent21 lppsource_ptx21
# nim -o define -t lpp_source \
  -a master=server \
  -a location=/spdata/sys1/install/lppagent22 lppsource_ptx22
# nim -o define -t lpp_source \
  -a master=server \
  -a location=/spdata/sys1/install/lppPTPE lppsource_ptpe
```

Then we used the following script to install the perfagent.server and ptpe.program filesets on a node:

```
echo $0: initialize

# set this up so that we can use dsh as well as pexec
nodes=$1
hostlist -n ${nodes} >/tmp/ptpe.inst.$$
export WCOLL=/tmp/ptpe.inst.$$

# next statement is needed for nimclient to rsh back to master
cp ${WCOLL} /.rhosts

# echo permit nim operations on node
dsh nimclient -o reset -F
dsh nimclient -p

# SOME debugging information
## for i in `cat ${WCOLL}`
## do
## lsnm -l $i
## done
## exit

echo $0: load latest perfagent.server on nodes
# for i in `cat ${WCOLL}`
# do
dsh nimclient -o cust -a lpp_source=lppsource_perf215 -a filesets=perfagent.server
$i
# done
# wait

echo $0: make a separate logical volume for data collection
dsh mklv -y lvPTPEdata rootvg 5
dsh crfs -v jfs -d lvPTPEdata -m /var/adm/ptpe -A yes -p rw -t no -a
frag=512 -a nbpi=16384 -a ag=8
dsh rm -f /var/adm/ptpe/*
dsh mount /var/adm/ptpe

echo $0: load latest ptpe.program on nodes
dsh touch /etc/perf/xmservd.res
# for i in `cat ${WCOLL}`
# do
dsh nimclient -o cust -a lpp_source=lppsource_ptpe -a filesets=ptpe.program $i
# done
# wait

echo $0: block further nim operations on node
dsh nimclient -P

echo $0: cleanup
rm ${WCOLL} /.rhosts

# echo rebuild bosboot on nodes
# pexec ${nodes} bosboot -a

echo $0: report from the nodes ${nodes}
pexec ${nodes} lsipp -l perfagent.server ptpe.program
```

Figure 97. Script to Install perfagent.server and ptpe.program on a Node

You will notice in Figure 97 that both `dsh` and `pexec` are being used. This is just to demonstrate how either of the commands could be used. Notice also the need for a `./rhosts` file to exist on the NIM boot/install server. In our case, this is the CWS; so we don't need to use `rcp` to copy `./rhosts` to supporting boot/install servers that may also be defined elsewhere on the SP.

7.4 Creating PTPPE Hierarchy Specifications

This section shows how you can create a PTPPE hierarchy that can be used in scripts and/or from the command line. For an example of creating a hierarchy with the Perspectives GUI, see 8.2.2, "Modifying PTPPE Hierarchy with Perspectives" on page 141.

The basic command syntax for creating a PTPPE monitoring hierarchy is:

```
ptpectrl -c <coordinator_host> -i <input_file
```

Note that the less than (<) character is shown bold above. That is because it is required to redirect the standard input (option `-i`). The argument `coordinator_host` should be the hostname of the node as stored in the SDR. In other words, if you are use fully qualified hostnames in the SDR, then your PTPPE hierarchy names must be fully qualified too. Compare Figure 98 on page 133 and Figure 99 on page 133 with Figure 102 on page 140.

Figure 98 on page 133 is a file that can be used as the input to the `ptpehier` command. The basis structure is:

```
{
name of first node in first group (is also data manager for group)
name of second node in first group
...
name of last node in first group
}
{
name of first node in a group (is also data manager for group)
...
name of last node in group
}
...
{
name of first node in last group (is also data manager of last group)
...
name of last node in last group
}
```

You may also choose to let `ptpehier` generate hierarchies for you. To generate a hierarchy based on Ethernet IP subnets, use the command:

```
ptpehier -e
```

To generate a hierarchy using frames as the basis for constructing the hierarchy, use:

```
ptpehier -f
```

When these commands are used, the coordinator is assigned at random. You may want to try these commands just to see what they do. The `ptpehier` command modifies information in the SDR.

```
{
n03_en0
n01_en0
n02_en0
n04_en0
}
{
n07_en0
n05_en0
n06_en0
n08_en0
}
{
n11_en0
n09_en0
n10_en0
n12_en0
}
{
n15_en0
n13_en0
n14_en0
n16_en0
}
```

Figure 98. Sample `ptpehier` Input File to Define a Hierarchy with 16 Nodes

The `ptpectrl -p` command causes the current monitoring hierarchy to be displayed:

```
[root]@cws1 > newgrp perfmon
[root]@cws1 > ptpehier -p
ptpehier: The current monitoring hierarchy structure is:
    n01_en0
        n15_en0
            n15_en0
            n13_en0
            n14_en0
            n16_en0
        n11_en0
            n11_en0
            n09_en0
            n10_en0
            n12_en0
        n07_en0
            n07_en0
            n05_en0
            n06_en0
            n08_en0
        n03_en0
            n03_en0
            n04_en0
            n01_en0

[root]@cws1 >
```

Figure 99. Hierarchy Displayed by `ptpehier -p`

7.5 PTPE Command Interfaces

There may be many reasons for choosing one interface above the other. Initially, we used the Perspectives interface to get a feel for what we were doing or, sometimes, trying to do. However, once we understood the basic mechanics of PTPE control, we used the command line interface more and more.

Table 5 is a summary of our thoughts. We are not trying to stress the usage of either interface. Your situation will probably be quite different than ours. We included it in this chapter because we feel you should be thinking about this while you are still making plans about how you are going to implement PTPE.

Table 5. The PTPE GUI vs. the PTPE Command Line Interface

| GUI Advantages | Command Advantages |
|---|---|
| Changes are not entered into SDR immediately. Further, user is given feedback when an incorrect hierarchy is attempted. | If you know what you want, and how to do it, the command line interface is faster - but always more error prone. |
| Provides graphical overview of current PTPE hierarchy that is probably more clear (in a report) for non-technical readers. | Can be used in scripts for manipulating the hierarchy. |
| Locks out PTPE commands for as long as you are busy, that assuring sole control. Drawback if you <i>forget</i> to close window when you are done. | Don't need to install an extra filesset. |
| Provides a good learning environment. You don't have to know the command line syntax to control PTPE. | Don't have to wait for the GUI to start. Don't need a graphical display to configure/operate PTPE data organization, collection and/or archiving. |
| Correct node names are entered in SDR, for instance fully qualified hostnames (if required). | Doesn't lock up the PTPE commands any longer than necessary. |

Note: On large SPs (for example 128 nodes) the Perspective takes considerably longer (about 10 minutes) than the command line interface (about 15 seconds).

7.6 Preparing PTPE Configuration Files

Actually, there are several different types of configuration files you need to be preparing, each for a different management domain.

The basic domains and configuration files are:

```

SDR      ptpehier command input
xmservd  xmservd.cf and xmservd.res
xmperf   xmperf.cf
PTPE     ptpe.cf

```

We described the PTPE hierarchy input formats above in 7.4, "Creating PTPE Hierarchy Specifications" on page 132. The layout for xmservd files is described in 4.1, "The xmservd Configuration" on page 73, and the layout for xmperf files is described in 2.3, "The xmperf.cf Configuration File" on page 14.

To see which statistics are being monitored, you can issue the `xmpeek` command as described in 4.5, “The `xmpeek` Command” on page 91.

```
> xmpeek -l
/f01n01/CPU/                Central processor statistics
/f01n01/CPU/gluser         System-wide time executing in user mode
...
/f01n01/DDS/              Dynamic Data Supplier Statistics
/f01n01/DDS/IBM/          IBM data suppliers.
/f01n01/DDS/IBM/PSSP.harmld/ Statistics provided by the harmld daemon
/f01n01/DDS/IBM/PSSP.harmld/CSS/ SP Switch statistics.
...
/f01n01/DDS/IBM/PSSP.harmld/LL/ LoadLeveler statistics.
...
/f01n01/DDS/IBM/PSSP.harmld/VSDdrv/ Virtual Shared Disk (VSD) statistics
...
/f01n01/DDS/IBM/XMservd/  XMservd Daemon Statistics
...
```

Figure 100. Using `xmpeek -l` to List PTPE Statistics Available

At the top of the list of statistics are the standard measures provided by the PTX agent, but further down are the additional elements provided by the PTPE reporter. The new items track information about the SP, including the switch, LoadLeveler performance, and the Virtual Shared Disk (VSD) environment.

There are additional statistics possible for a manager or coordinator node. These are the aggregate and summary data, respectively, that have been computed by the manager nodes and summarized by the central coordinator. Their location is under the `.../DDS/IBM/...` branch of the tree:

```
...
hosts/f01n03/DDS/IBM/PTPE_sum/DDS/IBM/...Aggregate: IBM-defined Dynamic Data
Suppliers
...
```

7.6.1 Controlling PTPE Data Collection

The `ptpe.cf` file is used to configure the communication between the coordinator, the managers and the reporters. Note that the extra statistics provided by PTPE can be obtained from any node on which either PTPE collection is made active (`ptpctr1 -c`) or the PTPE run-time command is running (`ptperrtm`). Any PTX Manager can access the DDS statistics through the Rsi and/or the Spmi API interfaces. PTPE data collection is a set of extra applications that are working together using the PTPE API.

In other words, `ptpe.cf` is the configuration file for a group of applications that communicate through the PTPE Application Programming Interface. The basic choices for the PTPE applications are (of all the statistics available through `xmservd` and DDS supplier programs): which statistics are to be reported to a manager who will average them and which statistics are to be archived for later offline analysis?

The layout of the `ptpe.cf` file is similar to `xmservd.cf`. The `ptpe.cf` file has an optional extra colon (`:`) followed by two digits, either a zero (0 meaning do not

collect/archive) or a one (1 meaning do collect/archive) separated by a comma (,). Thus:

statistic_name:collection_flag,archiving_flag

Some examples:

1. Mem/Virt/pagein
2. LAN/tok0/bytesout:1,1
3. Proc/runque:1,0
4. LAN/tok1/bytesout:0,1
5. CPU/glidle:0,0

Example 1 follows the default activity:report,archive. Example 2 also reports and archives. Example 3 is only reported at specified times to its manager, and example 4 is not reported to a PTPE manager, but will be archived. Example 5 is neither reported to a PTPE manager nor archived locally.

NOTE

You can set both collection and archive to off for a statistic here and then activate one or both later, during live monitoring, as needed.

Here is another example of ptpe.cf entries:

```
#token-ring statistics- collect & archive
LAN/tok0/bytesout:1,1
LAN/tok0/bytesin:1,1
#
#local runqueue- collect but don't archive
Proc/runque:1,0
#
#paging space- archive only
PagSp/*/%free:0,1
#
#disk busy- no collect, no archive
Disk/hdisk1/busy:0,0
```

Important

The statistics that are to be monitored are controlled through the xmservd configuration files. From these statistics, ptpe.cf configures which of these statistics is averaged and/or archived.

7.6.2 Enabling PTX for PTPE

To have xmservd automatically start the PTPE DDS, add the following line to /etc/perf/xmservd.res.

```
supplier: /usr/lpp/ptpt/bin/ptptrtm -p
```

The next time xmservd starts the program, ptpterm will also be started.

Sending a running xmservd the SIG_HUP signal (kill -1 <PID>) causes the (new) supplier programs to be started immediately.

7.7 Summary

We hope that the basic architectural building blocks are clear now. In review:

The basis for all statistics available are those delivered through `xmservd`. These statistics include any also provided by Dynamic Data Suppliers (DDS) such as `filtd` and/or `ptpertm`. Which statistics are made available are configured by `xmperf.cf`. The next layer of performance monitoring management is PTPE. The `ptpehier` command sets up the monitoring hierarchy; the `ptpectrl` command starts and stops the reporting and monitoring between reporters, (PTPE) managers and the coordinator. Besides the hierarchy stored in the SDR, the file `ptpe.cf` on each individual node determines the statistics that are reported and/or archived to the PTPE managers. Recall that the DDS statistics from `ptpertm` are always available to a PTX Manager regardless of the status of the PTPE monitoring programs.

Chapter 8. Using PTPE on SP Systems

This chapter covers the steps you, as a `ptpeuser` account, perform to use PTPE for collecting performance information on SP systems. The basic assumption of this chapter is that all software components have been installed and that the user has the proper security levels defined to use the PTPE (and PTX) subsystem. These topics are covered in Chapter 7, "Planning, Installing and Configuring PTPE" on page 119, and in the *Performance Toolbox Parallel Extensions for AIX: Guide and Reference*, SC23-3997.

All of the PTPE management functions can be performed from the command line. Starting with PSSP V2.2, there is an X-Window GUI interface called Perspectives (see Figure 103 on page 142). PTPE also has a fileset (`ptpe.gui`) for the Perspectives interface (see Figure 103 on page 142).

In this chapter, we demonstrate both interfaces for establishing a PTPE hierarchy.

We guide you through the steps required to make basic collection and archiving functions active on the SP system. Then we consider some sample scenarios that investigate common SP-related performance questions and show how specific information can be collected to help analyze problems.

In this chapter, we use the following system configuration for our tests:

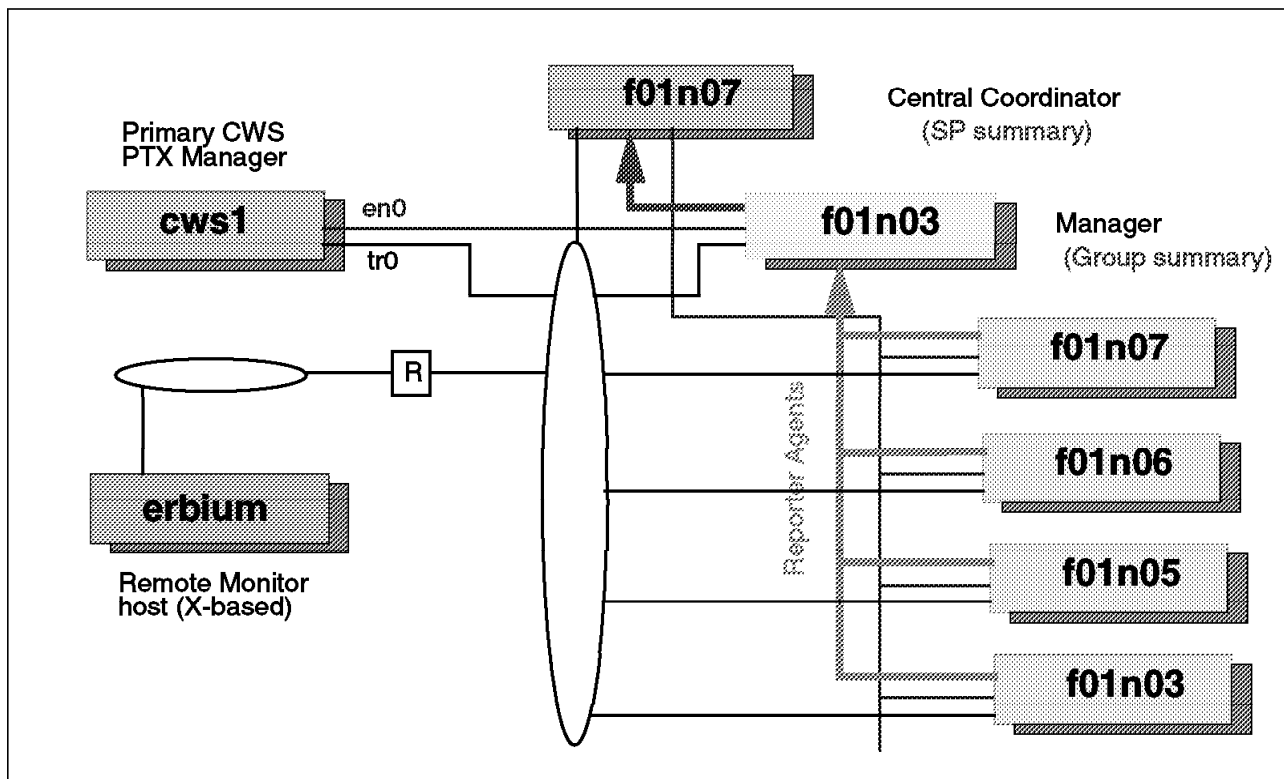


Figure 101. SP Environment for PTPE Testing

We have defined a user ID, `ptpeuser`, that has a primary group of `perfmon` to be our principal monitoring and control ID for PTPE. Our work is done remotely from `erbium`, an AIX host with a large display and 24-bit graphics adapter. This

is a typical arrangement for most SP installations, but many other variations are possible, especially with larger SP systems.

8.1 Verify PTPe Operational Status

This first step will verify that PTPe is available for use on the SP system, and will show if PTPe services for data collection and archiving are already active. Login to the SP using the `ptpeuser` account, and issue the `ptpectrl` command with the `-q` option.

```
f01n07:/u/ptpeuser > ptpectrl -q
ptpectrl: Performance information collection is not active.
ptpectrl: Performance information archiving is not active.
ptpectrl: Command completed.
```

8.2 Controlling PTPe Hierarchy

This step will verify that the current monitoring hierarchy defined to PTPe is indeed the one you want to use. Issue the `ptpehier` command with the `-p` option, to print the currently active hierarchy.

```
cws1:/ > newgrp perfmon
cws1:/ > ptpehier -p
ptpehier: The current monitoring hierarchy structure is:
    f01n03.sp.itsc.austin.ibm.com
        f01n01.sp.itsc.austin.ibm.com
            f01n07.sp.itsc.austin.ibm.com
            f01n06.sp.itsc.austin.ibm.com
            f01n05.sp.itsc.austin.ibm.com
            f01n03.sp.itsc.austin.ibm.com
            f01n01.sp.itsc.austin.ibm.com
cws1:/ >
```

Figure 102. Note the Fully Qualified Hostnames in this Output From `ptpehier`.

The output from this command shows that the central coordinator is `f01n03`, and `f01n01` is the data manager for a single group of five nodes. If this is not what you require, you may load a different hierarchy at this point by using the `-i` and `-c` options together with a new hierarchy specification file.

Note

Only one hierarchy can be used during PTPe collection/archiving. What this means is that the hierarchy cannot be adjusted, nor can the hierarchy be changed, while PTPe collection/archiving is running. If the hierarchy needs to be changed, PTPe collection/archiving needs to be shut down, then the hierarchy can be changed.

8.2.1 Modify PTPE Hierarchy with Command Line Interface

See 7.4, “Creating PTPE Hierarchy Specifications” on page 132, for how to build a custom hierarchy specification file.

```
f01n07:/u/ptpeuser > ptpehier -i -c f01n07.sp.itsc.austin.ibm.com <ptpe3.hf  
ptpehier: Monitoring hierarchy successfully created.
```

The `-i` option specifies that you will provide the hierarchy via standard input (which we have handled by redirecting stdin with our custom hierarchy file). This option also requires you to use the `-c` option to specify the new central coordinator node as an argument to the command.

Note

The host name specified for the new central coordinator must exist as a `reliable_hostname` in the Node class of the System Data Repository. The new hierarchy is then stored in the SDR.

Now that you have confirmed (or loaded) the right PTPE hierarchy definition, we need to check that the right `ptpe.cf` configuration file is in place and is supported by PTPE. As mentioned earlier, having several predefined configuration files available simplifies the collection of specific statistics for different activities. If you decide to use a custom configuration file with a name different than `ptpe.cf`, use the `-l` option to have PTPE read it instead. We can now use the `ptpectrl` command with the `-v` option to have PTPE validate the configuration file for us.

```
f01n07:/u/ptpeuser > ls ptp*.cf  
ptpe.cf  ptp1.cf  ptp2.cf  ptp3.cf  
..  
f01n07:/u/ptpeuser > ptpectrl -v -l $HOME/ptp3.cf  
ptpectrl: Configuration file appears valid.  
ptpectrl: Command completed.
```

8.2.2 Modifying PTPE Hierarchy with Perspectives

It is also possible to use the Perspectives Performance Monitor interface to perform the same functions using a graphical interface. To do so, start by authorizing remote X-clients to your X-server, login to the SP as `ptpeuser`, and export your `DISPLAY` variable. Start the Perspectives launch pad by entering the `perspectives` command.

```
erbium:/ > xhost cws1  
cws1 being added to access control list  
erbium:/ > telnet cws1  
login: ptpeuser  
ptpeuser's Password: xxxxxx  
..  
export DISPLAY=erbium:0  
perspectives
```

This brings up the main window of the Perspectives graphical user interface. You can then double-click on the **perfmon** tool with the left mouse button to bring up the Performance Monitor.

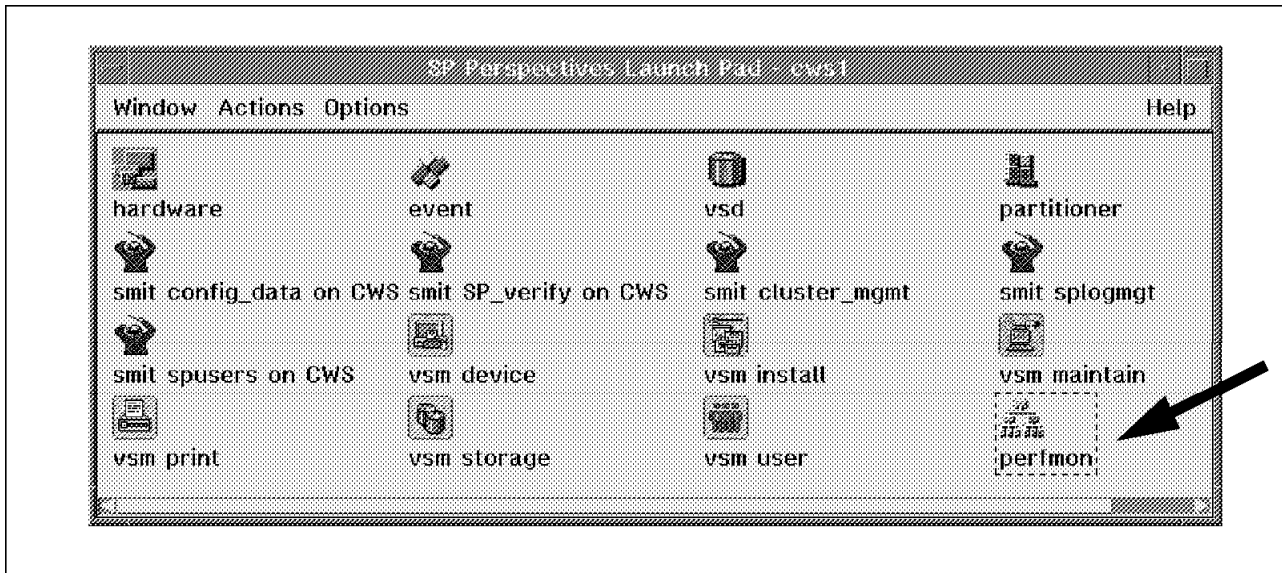


Figure 103. Perspectives Launch Pad with the Perfmon Tool in the Lower Right Corner

Note

It may take a moment for the Perspectives interface to appear. Please be patient while it checks the SDR, loads the Motif libraries, and reads profile and configuration data. If you only wish to use the Performance Manager tool, you may start it directly from the command line by using the command `spperfmon` instead of `perspectives`.

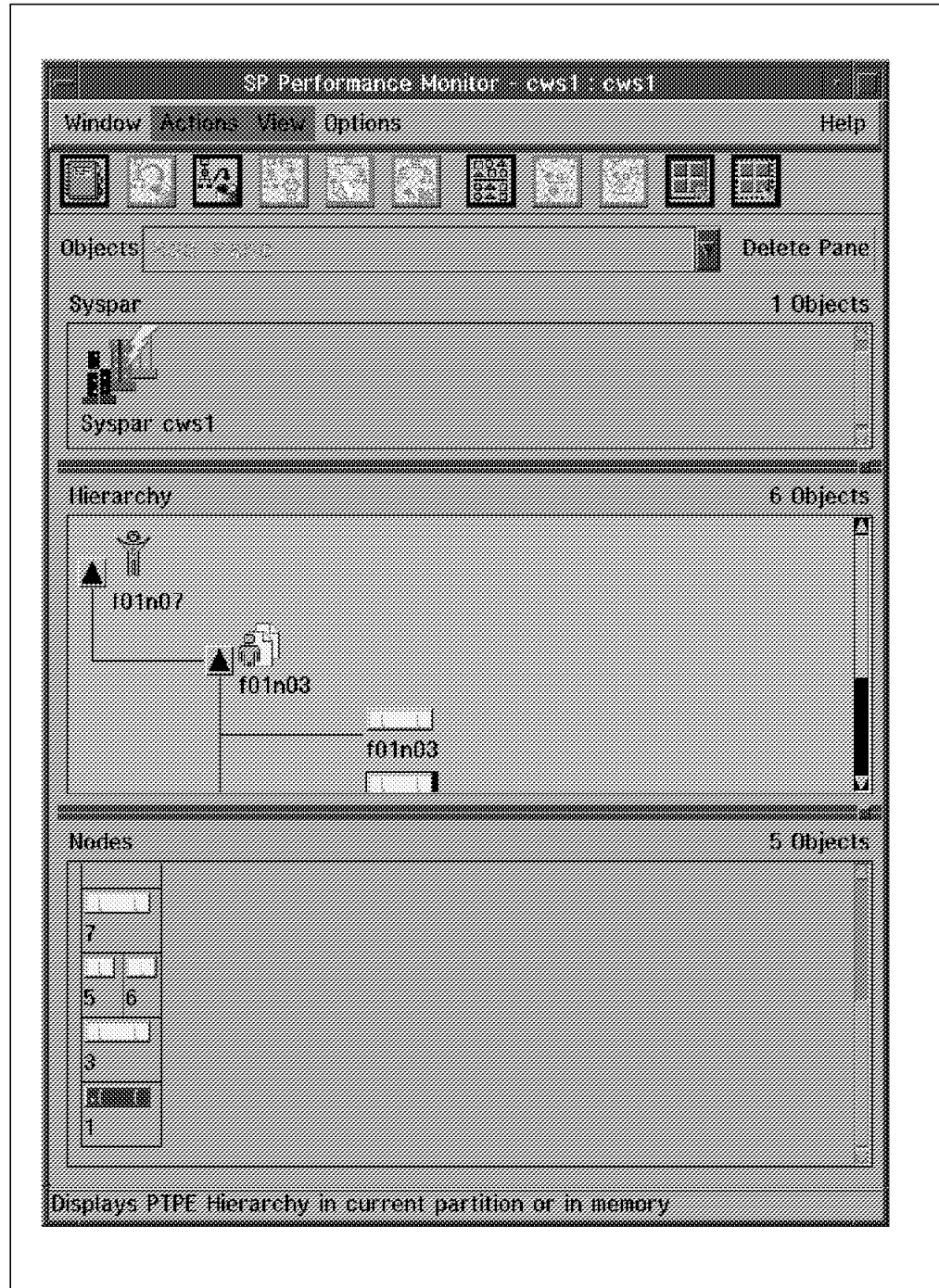


Figure 104. The Primary Perspectives Performance Monitor Interface

The middle panel of this display shows the current monitoring hierarchy with an indented tree format. It is composed of nodes selected from the Nodes panel at the bottom of the display. You may change this hierarchy by selecting a node with the left mouse button and then choosing an action from the Actions pull-down menus.

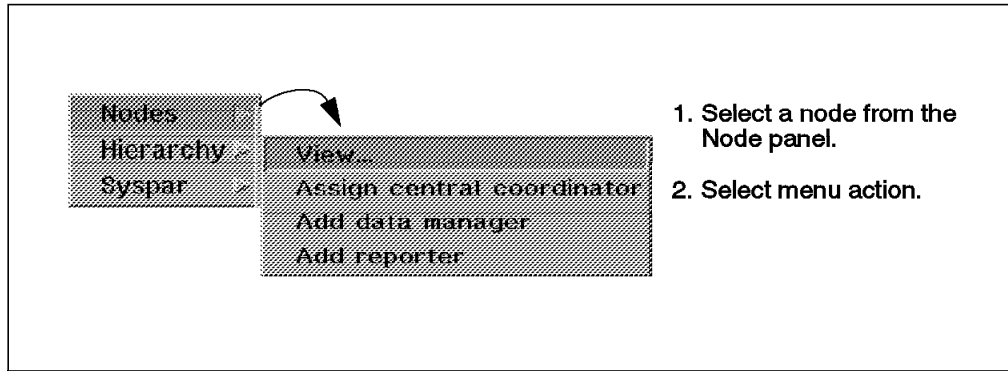


Figure 105. Node Options under Action Menu

When you select a member of the hierarchy panel, you can perform the following menu functions from the Hierarchy submenu.

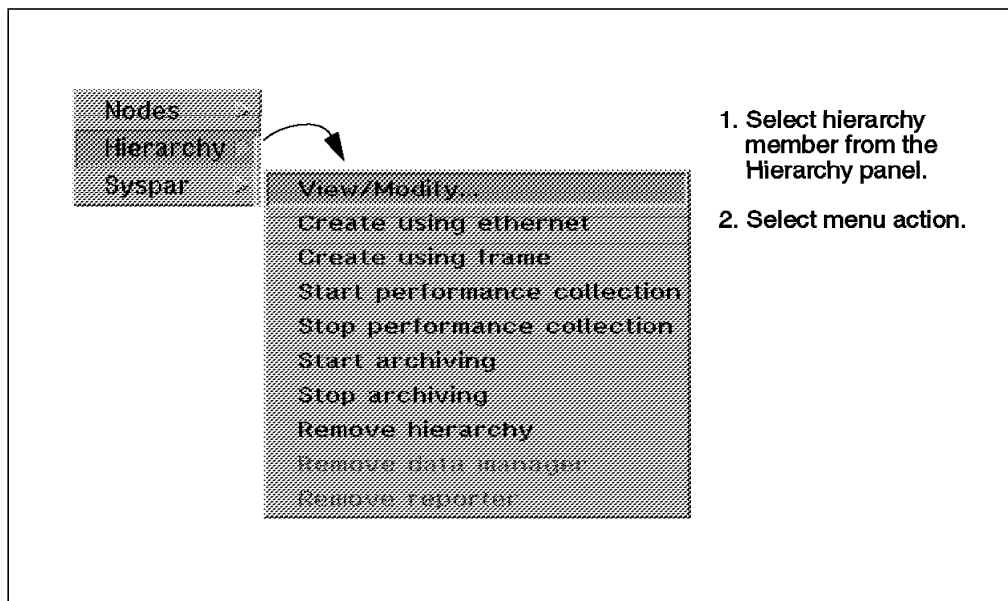


Figure 106. Hierarchy Options under Action Menu

After you have finished making any hierarchy changes with the Perspectives tool, you may save the final version to the SDR by using the **Save hierarchy to SDR** option under the Window pull-down menu. If you start to make hierarchy changes with the tool and then decide not to implement them, you can select the second option to revert back to the original hierarchy stored in the SDR.



Figure 107. SDR Options under the Window Menu

8.2.3 Initializing PTPE Hierarchy

With a valid hierarchy in place and the proper configuration file identified, we can now initialize PTPE on the system. As *ptpeuser*, issue the `ptpectrl` command with the `-i` option to request initialization. The PTPE master daemon, `spdm`, instructs all reporter nodes to discover available performance data and instructs `spdmcol` daemons to build aggregation and summary tables in shared memory. This process can take a little while to complete, depending on the number of nodes participating.

```
f01n07:/u/ptpeuser > ptpectrl -i
-----
ptpectrl: Beginning setup for performance information collection.
ptpectrl: Reply from the Central Coordinator expected within 170 seconds. OK.
ptpectrl: Setup for collecting performance information succeeded.
-----
ptpectrl: Command completed.
-----
```

An interval timer is started at 180 seconds, or with a new PTF at 270 seconds, as a watchdog to prevent problems from indefinitely hanging PTPE. If there is a problem with initialization on some members of the hierarchy, the errors are reported as shown below.

```
f01n07:/u/ptpeuser > ptpectrl -i
-----
ptpectrl: Beginning setup for performance information collection.
ptpectrl: Reply from the Central Coordinator expected within 155 seconds. OK.
ptpectrl: Setup for collecting performance information succeeded.
ptpectrl: 2516-026 System f01n05.sp.itsc.austin.ibm.com failed.
-----
ptpectrl: Command completed.
-----
```

In this example, the `f01n05` node was unable to initialize and was dropped from the group. This could be due to several causes, and more information can be obtained by reviewing the AIX syslog entries on the failing nodes. The central coordinator reports the initialization status of every node in the current PTPE hierarchy. When more than 50 percent of these nodes initialize properly, PTPE monitoring will begin.

Remember

Remember! PTPE starts collection as long as 50 percent or more of the reporting hierarchy initializes. If you are starting PTPE from within a script, checking only for command completion or return code zero may present a problem. If so, check for the 2516-026 message *System ... failed* to determine if any nodes encountered problems.

After PTPE has initialized successfully, we are ready to begin data collection. However, there is one more step that you may want to perform at this point. If you had run PTPE previously and been recording data to the performance data archives that you no longer want to keep, you can erase the archives on all nodes of the hierarchy with the `ptpectrl -e` command. This allows you to start a

new PTPE session with a clean set of archives. It is up to you to make a copy of the archives first, if there is any data that you wish to keep.

```
f01n07:/u/ptpeuser > ptpectrl -e
ptpectrl: Erasing the performance information archive files on all systems
          in the Performance Toolbox Parallel Extensions monitoring hierarchy.
          If the archive could not be erased on a system, the name of that
          system will be displayed below.
```

Note

Be sure that there are no other PTPE users that have archived data before you issue the command to erase the information. For example, a cron job may be recording snapshots every hour and then process the data archives over the weekend.

8.3 Controlling PTPE Data Collection

There are three basic operations we are concerned with here: starting, verifying and stopping PTPE data collection.

8.3.1 Starting PTPE Data Collection

To start data collection with PTPE, we can issue the `ptpectrl` command with the `-c` option. In addition, we will include the `-l` option to specify a particular configuration file to be used in place of the default, *ptpe.cf*.

At the time you start PTPE monitoring, you can also adjust the sampling frequency and archive rate from their normal default values. This is controlled by the `-f` option and uses two operands separated by a comma (*report,record*). The first specifies the number of seconds between data collection, and the second specifies the number of seconds between data archival. Both operands must be specified, and a zero may be used to indicate that no change is to be applied to a particular value.

The valid range for *report* is between 5 and 300 seconds, and the valid range for *record* is between 30 and 900 seconds. When selecting frequency values, remember that *record* must be evenly divisible by *report*. The default values are set at (15,60).

```
f01n07:/u/ptpeuser > ptpectrl -c -l ptp3.cf -f 15,60
-----
ptpectrl: Starting collection of performance information.
ptpectrl: Reply from the Central Coordinator expected within 155 seconds. OK.
ptpectrl: Performance information collection successfully started.
-----
ptpectrl: Enabling statistics for performance information collection.
ptpectrl: Restricting statistics from performance information collection.
ptpectrl: Enabling and restriction of statistics complete.
-----
ptpectrl: Attempting to modify performance information reporting and
          recording intervals
ptpectrl: Performance information reporting and recording intervals have been
          modified.
-----
ptpectrl: Command completed.
```

8.3.2 Verifying Proper Operation

Once PTPE data collection has started, it will continue running quietly as specified until either modified or shut down. To determine if data collection is currently active, you can issue the `ptpectrl` command with the `-q` option. This shows both the current operational status and the settings in use for the reporting and recording frequency variables.

```
f01n07:/u/ptpeuser > ptpectrl -q
ptpectrl: Performance information collection is active.
ptpectrl: Performance information archiving is not active.
ptpectrl: Current performance information reporting frequency is 15 seconds.
ptpectrl: Current performance information recording frequency is 60 seconds.
ptpectrl: Command completed.
```

At this point, PTPE data collection is active, and a limited set of statistics are being reported and summarized within the hierarchy as specified in the `ptpe3.cf` configuration file. The reporting frequency is set at 15-second intervals. You can now use any of the PTX tools or the PTPE API to analyze the reported data.

8.3.3 Stopping PTPE Data Collection

Stopping PTPE data collection is simple. The command is:

```
ptpectrl -s
```

8.4 Archiving Performance Data

Archiving data is an additional mode of data collection. The default mode is to report statistics to a data manager. Archiving may be activated as an alternate activity. Data is archived for analysis of, and/or reports on, performance measures.

Archiving may be started simultaneously with data reporting, later as part of data collection or set to record only when some event of interest occurs.

8.4.1 Starting Archiving and Collection Simultaneously

To start recording performance data at the same time you start collection, just add the `-r` option to the `ptpectrl` command. Note that the `-r` option must be specified after the `-c` option in the argument list.

```
f01n07:/u/ptpeuser > ptpectrl -c -r -l ptpe3.cf -f 15,60
-----
ptpectrl: Starting collection of performance information.
ptpectrl: Reply from the Central Coordinator expected within 165 seconds. OK.
ptpectrl: Performance information collection successfully started.
-----
ptpectrl: Starting archiving of performance information
ptpectrl: Performance information archiving successfully started.
ptpectrl: Enabling statistics for performance information collection.
ptpectrl: Restricting statistics from performance information collection.
-----
ptpectrl: Enabling statistics for performance information archiving.
ptpectrl: Restricting statistics for performance information archiving.
ptpectrl: Enabling and restriction of statistics complete.
-----
ptpectrl: Attempting to modify performance information reporting and
           recording intervals
ptpectrl: Performance information reporting and recording intervals have been
           modified.
-----
ptpectrl: Command completed.
```

8.4.2 Starting Archiving after Collection is Active

If PTPE data collection is already active and you now want to start recording to the Performance Data Archive, you can use the `ptpectrl` command with the `-r` option alone or together with the `-f` option to control just the archive activity. This capability is often used when PTPE is running with data collection only (for example to feed real-time PTX consoles), and you want to record performance data briefly for some system event.

```
f01n07:/u/ptpeuser > ptpectrl -r
-----
ptpectrl: Starting archiving of performance information.
ptpectrl: Reply from the Central Coordinator expected within 180 seconds. OK.
ptpectrl: Performance information archiving successfully started.
-----
ptpectrl: Enabling statistics for performance information archiving.
ptpectrl: Reply from the Central Coordinator expected within 180 seconds. OK.
ptpectrl: Restricting statistics for performance information archiving.
ptpectrl: Reply from the Central Coordinator expected within 180 seconds. OK.
ptpectrl: Enabling and restriction of statistics complete.
-----
ptpectrl: Command completed.
```

8.4.3 Stopping Archiving without Stopping Collection

Once PTPE data archiving tasks are complete, you can stop the archiving process by issuing the `ptpectrl` command with the `-t` option. This closes the archives on all nodes in the current monitoring hierarchy and makes the newly archived data available for use. Normal data collection and summarization activities continue to remain active within the hierarchy.

```
f01n07:/u/ptpeuser > ptpectrl -t
-----
ptpectrl: Reply from the Central Coordinator expected within 180 seconds. OK.
ptpectrl: Performance information archiving successfully stopped.
-----
ptpectrl: Command completed.
-----
```

8.5 Displaying Selected Statistics with Online PTX Consoles

In this section, we see how to view the performance information being collected and summarized by PTPE across the hierarchy.

Typically you will use the PTX consoles (xmperf, 3dmon) to view the average statistics calculated by the PTPE data manager nodes (those in the DDS/IBM/PTPE_sum/* subtree) instead of the statistics from each of the nodes. This was the reason for building PTPE in the first place: to reduce the amount of data that needs to be monitored in order to determine the performance status of the system. Instead of monitoring the nodes themselves (especially larger SPs, like 32 - 128 node configurations), you monitor a subset of the nodes to determine what is going on in the system. If the averages start to appear unusual, then you can access the individual nodes within the manager's group, using the PTX consoles to find the nodes that are having problems. But you don't need to monitor all the nodes all the time. This is a point that needs to be stressed. It is the added value of PTPE.

PTPE provides additional statistics to the Performance Toolbox (PTX) Manager. The only new area is defining new consoles that will monitor the PTPE data collection manager(s) and/or the PTPE coordinator. Rather than repeat a great deal of information covered in Part 1, we discuss ways in which you can organize your configuration files for ease of use.

The tools used are familiar from the previous chapters on PTX. We focus primarily on the xmperf consoles and the 3dmon displays and show how the PTPE data elements can be integrated to gain an overall view of activity within the SP system.

We suggest organizing your configuration files in a hierarchy of directories. The initial directory will have configuration files set up to monitor the PTPE coordinator and PTPE data managers. The default host is likely to be the PTPE coordinator. The Rsi.hosts file lists only these nodes and possibly the CWS for an overall view of the SP performance status.

From this (PTPE hierarchy home) directory, extra directories are created, each containing configuration files targeted for the data manager and its reporting nodes. In this directory, the Rsi.hosts file contains only the host names or addresses of the PTPE data manager and the PTPE reporting nodes. You may also include the coordinator, but the focus should be on the reporting group.

In all of these directories, the following files should be established:

- Rsi.hosts
- xmperf.cf
- 3dmon.cf

In the hierarchy home directory, the following may also be defined:

- exmon.cf

We use the following example ptp3.cf configuration file for PTPE, tailored to limit data collection and archiving to a selected subset of variables.

```
# Start Of Example Section
#-----

# Collect and Archive for all switch variables
DDS/IBM/PSSP.harmlD/CSS/*:1,1
#
# Skip the LoadLeveler items since LL is not active
# DDS/IBM/PSSP.harmlD/LL/*:0,0
#
# Collect but don't Archive the XMserve daemon statistics
DDS/IBM/XMservd/*:1,0
#
# Collect and Archive the token-ring and ethernet statistics
LAN/tok0/*:1,1
LAN/ent0/*:1,1
#
# Collect and Archive all CPU measures
CPU/*:1,1
```

8.5.1 Using xmperf Consoles

The most common method for viewing collected data from PTPE is with the PTX xmperf tool. By defining a console containing the desired set of indicators, you can quickly review the current status of the monitoring hierarchy. You may also define additional consoles that allow you to take a closer look at certain statistics when circumstances warrant. Use exmon to monitor thresholds for starting monitors of reporting groups giving greater detail.

Defining consoles is introduced in 2.3.2, "The xmperf Console" on page 17.

The sample configuration files that ship with PTPE include xmperf.cf_ptpe, an extension to the baseline xmperf.cf file that adds new capabilities to xmperf when used in conjunction with PTPE on an SP system. When integrated with the normal xmperf configuration statements, a new series of menu items become available under the Utilities menu bar option of the main xmperf window:

- Performance Toolbox Parallel Extensions:
 - PTPE Status Report
 - Display Monitoring Hierarchy
 - Initialize Monitoring Hierarchy
 - Begin Performance Data Collection
 - Begin Performance Data Archiving
 - Change Sampling and Archiving Rates
 - Stop Performance Data Archiving
 - Stop Performance Data Collection

By using these xmperf menu items, you can accomplish most tasks associated with managing PTPE functions right from the xmperf environment. These actions invoke the relevant PTPE commands in a subshell and display the results in a separate aixterm window on the screen.

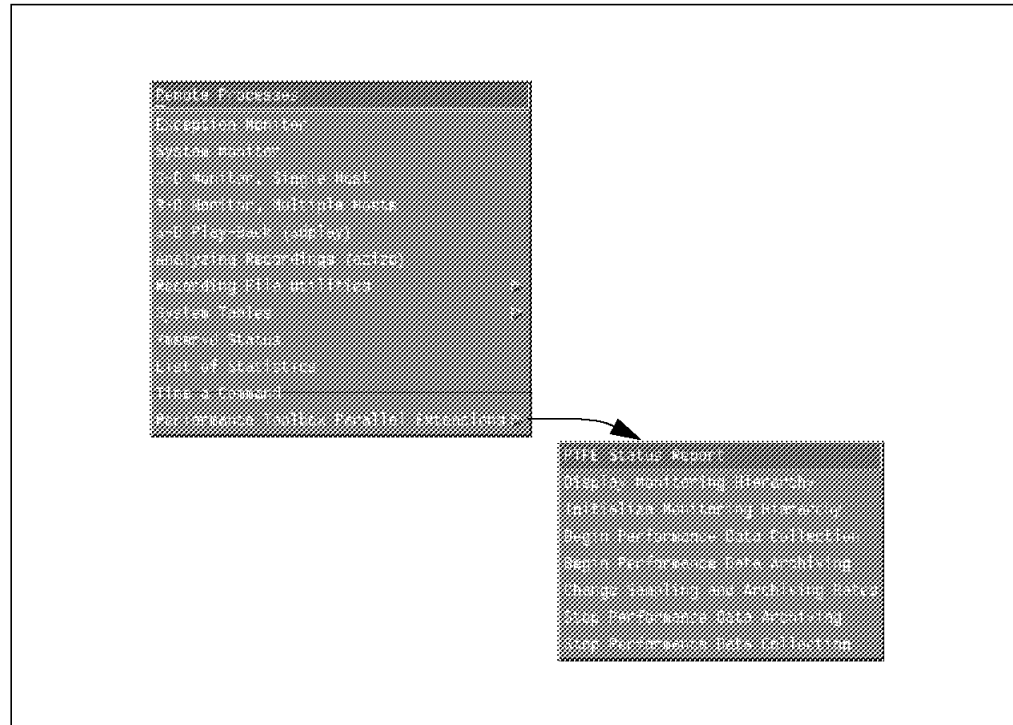


Figure 108. New Options under the Utilities Menu after Integrating PTPE

In addition, there are several new consoles added to the configuration file that can be found under the Monitor menu of the main xmperv window.

PTPE Data Manager Mini Monitor

A skeleton console similar to the local system mini monitor. This console can monitor summary statistics on any manager node selected by the user.

PTPE Data Manager Summary Monitor

A skeleton similar to the local system monitor. This console can monitor summary statistics on any manager node selected by the user. The tty I/O variables have been replaced with Switch statistics.

PTPE Data Manager Combo Monitor

A skeleton similar to the local system combination monitor. The local system statistics have been replaced with summary statistics available from any manager node selected by the user.

PTPE Local Switch Monitor

A local system console that monitors Switch performance information on the local host.

PTPE Remote Switch Monitor

A skeleton console that monitors the same information as the PTPE Local Switch Monitor, but on any node selected by the user.

PTPE Local VSD Monitor

A local system console that monitors Virtual Shared Disk performance information on the local node.

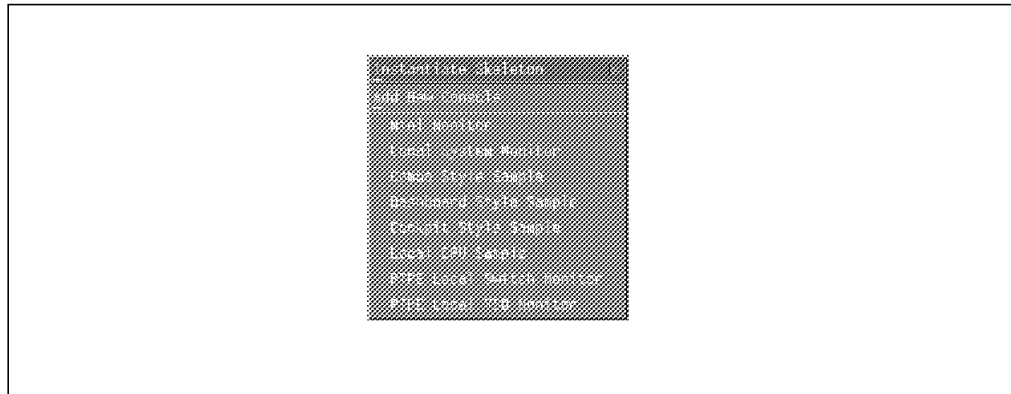


Figure 109. New Consoles Added to the Monitor Menu

8.5.2 Using 3dmon Displays

Another common method of viewing PTPE data is with the PTX 3dmon tool. This application provides a unique pseudo-3-D view of Z-axis value data across X and Y dimensions. This can be very helpful in getting a high-level comparative overview of multiple system elements in an SP environment.

In order to help PTPE users get started quickly, IBM includes a useful group of configuration sets that can be merged into existing 3dmon.cf configuration files. This provides a new set of 3dmon displays that can show summary data reported from manager nodes in several different groupings.

The following configuration sets are available in *3dmon.cf_ptpe* in the */usr/lpp/ptpe/samples* directory:

- | | |
|------------|---|
| PTPE_Hosts | This configuration set is a copy of the default configuration set used by 3dmon. The statistics have been changed to monitor the summary statistics provided by PTPE Data Manager and Central Coordinator nodes. The user is prompted for the hosts to be monitored. |
| PTPE_24 | This configuration set is a copy of the 24 configuration set found in 3dmon.cf. The statistics have been changed to monitor the summary statistics provided by PTPE Data Manager and Central Coordinator nodes. The user is prompted for the hosts to be monitored. |
| PTPE_disk | This configuration set is a copy of the disk configuration set found in 3dmon.cf. The statistics have been changed to monitor the summary statistics provided by PTPE Data Manager and Central Coordinator nodes. The user is prompted for the hosts and disks to be monitored. |
| PTPE_kmem | This configuration set is a copy of the kmem configuration set found in 3dmon.cf. The statistics have been changed to monitor the summary statistics provided by PTPE Data Manager and Central Coordinator nodes. The user is prompted for the hosts and kernel memory areas to be monitored. |
| PTPE_lan | This configuration set is a copy of the lan configuration set found in 3dmon.cf. The statistics have been changed to monitor the summary statistics provided by PTPE Data Manager and Central |

| | |
|---------|---|
| | Coordinator nodes. The user is prompted for the hosts and LAN adapters to be monitored. |
| PTPE_fs | This configuration set is a copy of the fs configuration set found in 3dmon.cf. The statistics have been changed to monitor the summary statistics provided by PTPE Data Manager and Central Coordinator nodes. The user is prompted for the hosts and file systems to be monitored. |
| PTPE_ip | This configuration set is a copy of the IP configuration set found in 3dmon.cf. The statistics have been changed to monitor the summary statistics provided by PTPE Data Manager and Central Coordinator nodes. The user is prompted for the hosts and network interface to be monitored. |
| vsddrv | This configuration set permits 3dmon to display the performance information tracked by the Virtual Shared Disk driver. The user is prompted for the host to be monitored. |
| vsd | This configuration set permits 3dmon to display the performance information tracked for any Virtual Shared Disk. The user is prompted to select both the host systems and the Virtual Shared Disk devices to be monitored. |
| css | This configuration set permits 3dmon to display the performance information supplied by the SP Switch device. The user is prompted for the host to be monitored. |

8.6 Offline Analysis of Archived Data

In addition to real-time monitoring of live data collected by PTPE, another common function performed by SP administrators is the analysis of data collected during previous sessions and held for subsequent processing. This could include short duration snapshots of unusual system activity or longer collections of normal processing periods that need to be summarized and reviewed for workload trends prior to archiving. In both cases, the activities necessary to accomplish this can be broken into three main areas.

1. Extract the relevant information from the PTPE archive files.
2. Convert the data to a format suitable for the tool(s) you plan to use.
3. Process the data with a display playback tool or application program.

Which methods you use to accomplish these tasks can vary depending on the intended objectives. We cover the more typical ones below and show how other variations are possible by using different options.

8.6.1 Using `ptpedump` to Build ASCII files

The `ptpedump` command can be used to have PTPE read the binary performance data archive on selected nodes and produce an ASCII text formatted version of their entire contents. The exact form of the output can be controlled by several switches that allow the resulting output to be used with data conversion tools, imported into a spreadsheet or used as input to load database tables.

```
f01n07:/u/ptpeuser > ptpedump -n f01n03 f01n05 f01n06
ptpedump:
ptpedump: command has completed
```

If a subset of nodes is not specified, the ptpedump command retrieves the current monitoring hierarchy from the SDR and invokes a distributed shell on each member to run the spdm_dump utility. Otherwise, only the list of nodes specified are instructed to begin dumping their performance archive data.

Note

The current monitoring hierarchy in the SDR may not correspond to the set of nodes active during data collection if the hierarchy was later modified. You must insure that the SDR hierarchy still matches that of the data collection period, or specify the equivalent set of nodes with the -n option.

The performance data archive located in /var/adm/ptpe/ is read sequentially, and the extracted data is written back into the same directory. Therefore, you must insure that there is adequate space on all nodes to contain the resulting output. If space is insufficient on a node, you can run the spdm_dump utility manually and redirect standard out to another filesystem with enough space to contain the output.

The output format of the ASCII text can be varied by using the proper command switches. By default, ptpedump produces a single line per observation in the following format.

Timestamp Length Data Type Statistic Value

When the -c option is specified, a comma-delimited format is used instead:

Time="Timestamp", Statistic=Value

When the -s option is used, a wide (132 columns) spreadsheet style is used, with the statistic names along the horizontal axis and time along the vertical.

By selecting the proper option for formatting, you can simplify the amount of manipulation required in subsequent steps and insure that consistent formats are maintained.

8.6.2 Using a2ptx to Create Recording Files

Once you have extracted the binary data from the performance data archives and converted it into default ASCII format, you have the option of converting it to a format that PTX tools can accept as a recording file. The a2ptx utility provided with PTX can be used to read the ASCII output files produced by the ptpedump command. The output can then be used as input to any recording support program, including xmperf, 3dmon or azizo.

8.7 What to Do When a Node Goes Down?

If you suspect that a node is down, do the following:

```
cws1:/u/ptpeuser > ptpectrl -s
-----
ptpectrl: Reply from the Central Coordinator expected within 105 seconds. OK.
ptpectrl: 2516-037 Could not stop collecting performance information.
ptpectrl: 2516-025 Manager node f01n01.sp.itsc.austin.ibm.com failed.
cws1:/u/ptpeuser > ptpectrl -q
ptpectrl: Performance information collection is active.
ptpectrl: 2516-052 Performance information archiving is currently in Error
state. Please take corrective action.
ptpectrl: Current performance information reporting frequency is 15 seconds.
ptpectrl: Current performance information recording frequency is 0 seconds.
ptpectrl: Command completed.
cws1:/u/ptpeuser >
```

Checking the documentation, we found only the following command. Because there was only one manager, and it was down (we stopped it), there was no one to receive the command. So a variation of the error was returned. The documentation states that you should also check the status of the SDR entries.

```
cws1:/u/ptpeuser > ptpectrl -t
-----
ptpectrl: 2516-032 Performance information archiving is not active. The action
you have requested cannot be carried out unless performance information
archiving is active.
ptpectrl: 2516-039 Could not stop archiving performance information.
```

We waited for the node to be active again before continuing.

```
cws1:/u/ptpeuser > ptpectrl -q
ptpectrl: Performance information collection is active.
ptpectrl: 2516-052 Performance information archiving is currently in Error
state. Please take corrective action.
ptpectrl: Current performance information reporting frequency is 15 seconds.
ptpectrl: Current performance information recording frequency is 0 seconds.
ptpectrl: Command completed.
cws1:/u/ptpeuser > ptpectrl -s
-----
ptpectrl: Reply from the Central Coordinator expected within 180 seconds. OK.
ptpectrl: Performance information collection successfully stopped.
-----
ptpectrl: Command completed.
-----
```

And we are back at a known situation.

Unfortunately, recovering from a node failure is not always this simple. There are two kind of failures to consider:

1. Daemon failure

A daemon failure is relatively easy to recover from. In shutting down PTPE with `ptpectrl -s`, the command will detect that the daemon is already down. It will consider this an acceptable condition (after all, it would have shut the daemon down anyway) and continue to shut down all other daemons. You can then try to start PTPE again and examine the error messages.

2. Node failure

Node failure is an entirely different situation. The shutdown command cannot contact the node, and does not know why it cannot contact the node. Therefore, it records an error status in the SDR. Now, if this node is simply a reporter, then the recovery is easy. If the node is a data manager, it is harder (a failure in the Central Coordinator is the toughest). Basically, the PTPE daemons that report up the hierarchy (for instance, the daemon that reports to its manager node, and the manager that reports to the coordinator) will detect when the node, it reports to, fails. If a failure to report fails 5 times in a row, the daemon that was trying to send data up the hierarchy considers its superior to have failed, and shuts itself down. So, if the collection interval is 15 seconds, a reporter will consider its manager down if it cannot send data for 1 minute 15 seconds; after 75 seconds of time, all reporters in this group should be shutting themselves down, give or take a few seconds. This also applies to managers reporting to the central coordinator. So, in the case of a Central Coordinator failure, using the same collection interval, all managers will shut down in about 75 seconds, and all reporters will shut down 75 seconds after that. Problem is: the SDR still shows collection as active. This will prevent other PTPE commands from working, including the `ptpehier` command to remove the failed node from the hierarchy until the node is brought back online. This is actually designed this way, because we didn't want the user to alter the hierarchy while collection was either active or in error state. This assists in locating and debugging the problem (and to prevent the reporting relationships from getting confused on the nodes). So, what's the recovery?

- a. Try to shut down PTPE collection/archiving with `ptpectrl -s`.

This will shut down as much of the daemons as possible. If a manager node is down, then it will be impossible for the command to shut down its reporters (remember, the commands are coordinated by the managers); if the Central Coordinator is down, none of the nodes will get the command.

- b. Verify that all the PTPE daemons are inactive, using a `ps` command.

The easiest way is through `dsh` from the control workstation:

```
ksh prompt> dsh -av ps -ef | grep spdm
```

If any nodes show any `spdm*` daemons active, go to those nodes and kill the related process (`kill <pid_number>`). Do not use `kill -9`, since this will prevent the daemons from cleaning up their shared memory and temporary files, and possibly prevent the next instance of the daemon from running correctly.

- c. Clear the collection status in the SDR:
`SDRChangeAttrValues SPDM active=0`
- d. Use `ptpehier` or `Perspectives` to remove the failed node from the hierarchy.
- e. Restart collection/archiving with `ptpectrl` or `Perspectives`.

Part 3. Monitoring and Analysis

Chapter 9. Monitoring and Analysis Hints

As you begin to form your monitoring strategy, you need to determine how broad a view of your performance situation is necessary. Do you want to take occasional glances at how the system is behaving, or do you want the ability to look at a period of time and get exact numbers on system performance?

If you have recorded information, do you have tools to analyze the data? Some tools are included with the manager code (`exmon`, `xmperf`, `3dmon`); others are packaged with the agent code (`ptxtab`, `ptxsplrit`, etc.). Still other useful tools for data analysis, like spreadsheets, `perl` or statistical analysis programs such as SAS, are not provided with Performance Toolbox.

9.1 Monitoring

Monitoring in PTX can be executed in several different ways:

- With the `exmon` command.
- With the `3dmon` command.
- With the `xmperf` command.
 - IP response time
 - Application response time
 - Lotus Notes servers

9.1.1 Monitoring with `exmon`

The exception program `exmon` is designed to provide a convenient facility for monitoring exceptions as they are detected on both the local hosts and remote hosts. The exceptions are generated as a result of alarm conditions defined in `filter.cf` and triggered by `filtd`.

The layout of the `exmon` monitoring window is that of a matrix with eleven column headings and a variable number of rows, each with a hostname identifier. In this figure, the EXmon resource file has been modified to include some more meaningful column headings. Please refer to 3.3.1, "The EXmon Resource File" on page 66, for further details in configuring `exmon`.

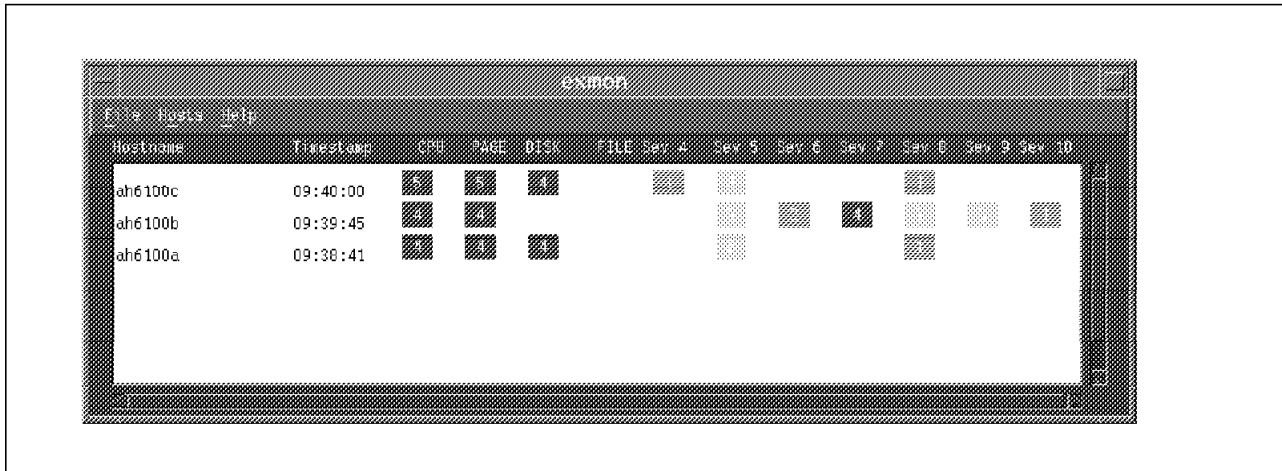


Figure 110. The exmon Window

The timestamp is that of the last exception received by that host.

The cells are assigned a color depending on the coloring scheme specified in the resource file. See 3.3.2, "The exmon.cf Configuration File" on page 69, for information on modifying the default coloring scheme.

As exceptions arrive, the matrix starts to become populated. The matrix cell corresponding to the hostname and the exception ID contain the count of exceptions. With each exception, a line is added to the host exception file for that host. The host exception files are stored in \$HOME/FilterLogs, as hostname.log.

To view the exceptions, select **Read Log** from the **File** pull-down menu. A file selection window is displayed with all the available log files.

After selecting your desired log file, the following window is displayed:

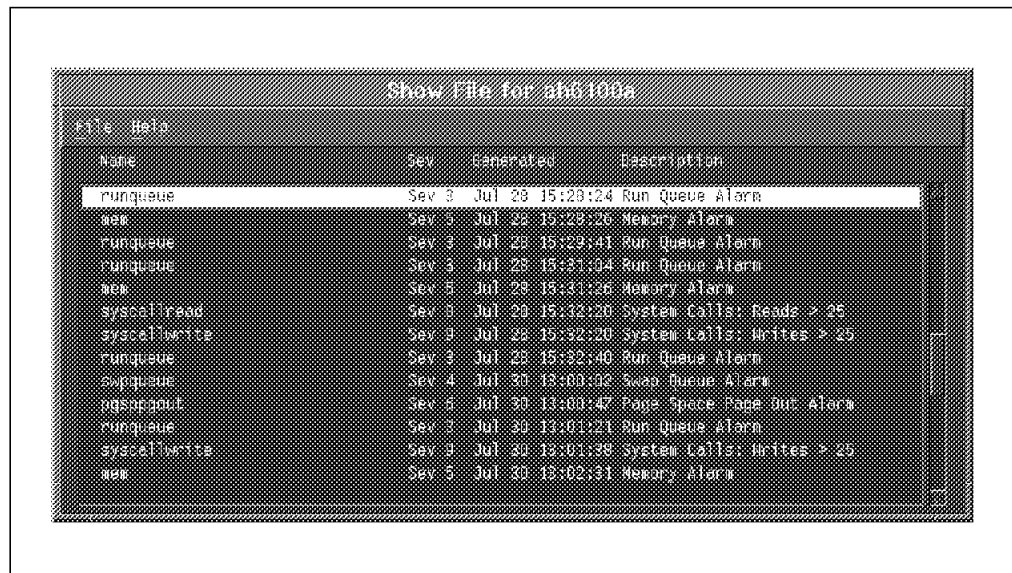


Figure 111. The exmon Show File Window

The File pull-down menu will allow you to:

- Mark for Delete** When the user saves the changes and closes this window, the selected lines are deleted from the exception log file.
- Mark for Read** A @ will appear at the beginning of the line. When the user saves and closes the window, the selected items are re-written to the log with a @ at the beginning of the line. The next time the user views the exception log, the @ will denote the exception has already been viewed.
- Save & Quit** This will re-write any entries that have been marked as read and delete any items marked for deletion. The Show File window will also be closed.
- Quit** The Show File window will be closed, but any entries that have been marked will not be modified.

By selecting a hostname with the left mouse button in the exmon main window, the user has the ability to execute commands that provide additional information about that system.



Figure 112. The exmon Command Execution Window

Refer to 3.3.2, "The exmon.cf Configuration File" on page 69, for information on customizing this window.

9.1.2 Monitoring with 3dmon

The 3dmon program, when started with the *lanresp* configuration set, displays response time in a matrix showing response time in both directions. The right side of the matrix represents the monitoring hosts, and the left side represents the monitored hosts. The measuring of response time in both directions allows for detection of situations where two hosts use a different route to reach each other, and it can also be used to pinpoint other network problems.

The 3dmon.cf file distributed in */usr/samples/perfmgr* contains the configuration set named *lanresp* for monitoring the IP network response time. The code is below. This code prompts the user for the desired host(s) to monitor and then executes the resptime statistic, which monitors the host(s) as well.

```
wildcard: lanresp hosts # response time between multiple hosts
RTime/LAN/*/resptime
```

Now let's run the *lanresp* configuration set.

Example:

```
3dmon -f /usr/samples/perfmgr/3dmon.cf -c lanresp
```

You are first prompted with the host(s) selection screen; let's select the first three hosts:

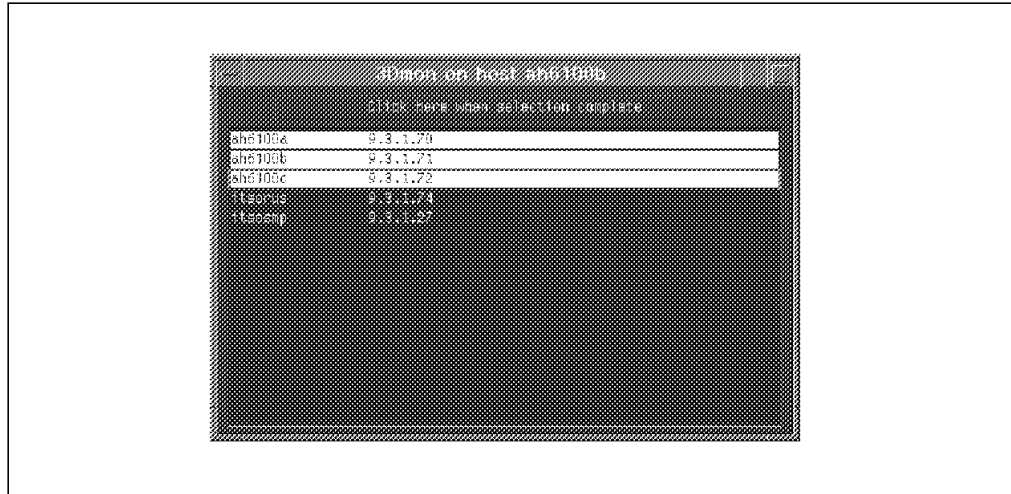


Figure 113. 3dmon Host Selection

After pressing **Click here when selection complete**, the following appears on your screen:

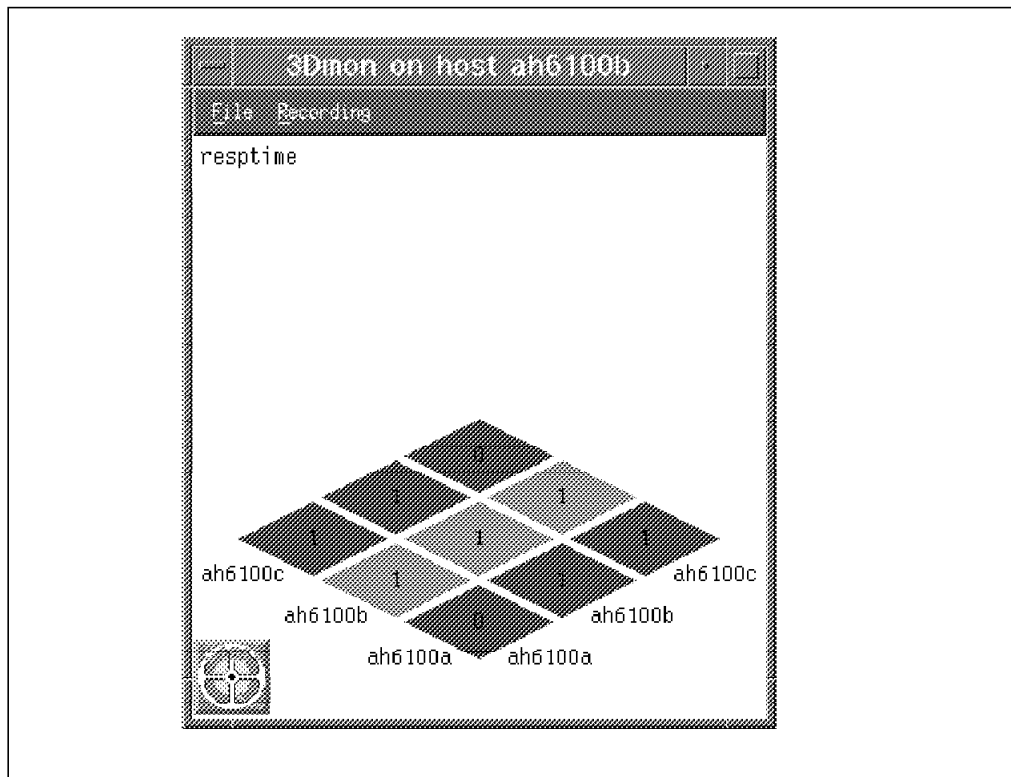


Figure 114. 3dmon lanresp Configuration Set

Because of the way 3dmon monitors IP response time, all hosts must be running xmservd. Only hosts responding to the invitation from 3dmon will appear in the hosts selection window.

9.1.3 Monitoring with xmperv

The xmperv program allows the user to monitor using the default values in the Monitor or Instantiate Skeleton pull-down. Refer to Figure 115. You also have the option to create a new monitor using the **Add New Console** option. These statistics come in two main forms: State graphs refer to 2.3.6.4, "State Graphs" on page 31, and recording graphs refer to 2.3.6.3, "Recording Graphs" on page 30. Depending on your need, you may select or create from either.

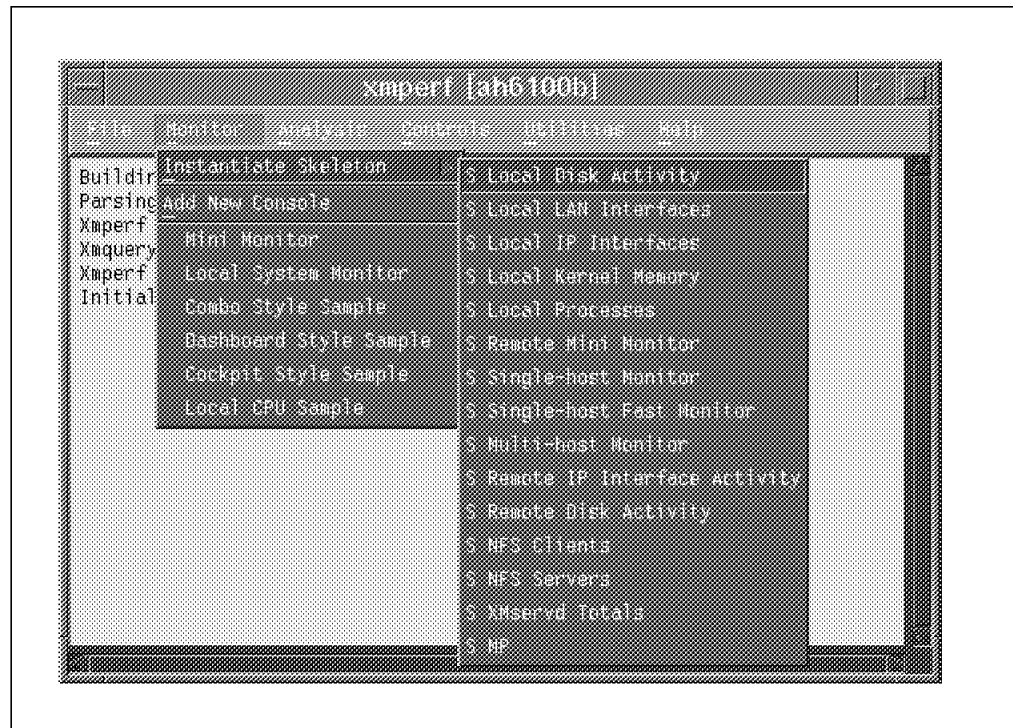


Figure 115. The xmperv Monitor Default Options

9.1.3.1 Monitoring IP Response Time

The xmperv program allows the user to monitor IP response time from two situations:

1. When the user displays the value selection window for the context RTime/LAN.
2. When the user instantiates a skeleton console that refers to IP response time measurements.

In both cases, the list of hosts to monitor are those whose xmservd daemons responded to an invitation from xmperv as well as hosts for which contexts were added by means such as the iphosts program. See 4.3.2, "The iphosts Command" on page 89, for more information.

The following values can be monitored;

- resptime** The latest observed time in milliseconds.
respavg The weighted average response time in milliseconds.

| | |
|------------------|--|
| below10 | The percentage of observations of response time that were less than 10 milliseconds. |
| below20 | The percentage of observations of response time that were less than 20 milliseconds. |
| below100 | The percentage of observations of response time that were less than 100 milliseconds. |
| above99 | The percentage of observations of response time that were at 100 or more milliseconds. |
| requests | A counter value giving the number of Internet Control Management Protocol (ICMP) requests sent to the hosts. |
| responses | A counter value giving the number of ICMP requests received from the hosts. |

Note

When creating instruments to monitor IP response time, remember that all statistics within an instrument must come from the same host.

9.1.3.2 Monitoring Application Response Time

Only those applications that have specifically been instrumented to provide Application Response Measurement (ARM) can be monitored. See 4.4.1, "The SpmiArmd.cf File" on page 90, for instructions on enabling the SpmiArmd daemon. Also, refer to *Performance Toolbox for AIX Guide and Reference, SC23-2625, Chapter 15* for details on the ARM subroutines.

Once the application is instrumented, the following statistics are available for each transaction.

| | |
|-----------------|---|
| resptime | The last measured response time for successful transaction in milliseconds. |
| respavg | The weighted average response time for successful transactions in milliseconds. |
| count | The number of successful transactions. |
| aborted | The number of aborted transactions. |
| failed | The number of failed transactions. |
| respmax | The maximum response time for successful transactions in milliseconds. |
| respmin | The minimum response time for successful transactions in milliseconds. |

9.1.3.3 Monitoring Lotus Notes Servers

The PTXNOTES Agent can be used to graphically monitor AIX Notes servers from a single graphical console. The Agent runs as a background daemon from a Notes Add-In task and consumes very little system resources (< 1% CPU). It provides real-time statistics on a variety of Notes and operating system metrics. The tool could be very useful for a Notes system administrator for monitoring and optimizing a Notes network and for troubleshooting potential problems. It also has the capability to record data for later playback and analysis. Alarms and triggers can be set to warn an operator when certain preset levels are reached.

The following types of statistics can be monitored:

- Notes Web_retriever Stats
- Notes Database Stats

- Notes Mail Stats
- Notes Server Stats
- Domino Server Stats
- Notes Memory Stats

The following is a list of supported statistics for PTXNOTES. This is in addition to the standard suite of operating system and network statistics that are also available.

Notes Web_retriever Stats

| | |
|-------------------|--|
| WebAccessFtp | Number of FTP retriever accesses |
| WebAccessGopher | Number of Gopher retriever accesses |
| WebAccessHttp | Number of HTTP retriever accesses |
| WebBytesReceived | Number of retriever bytes received |
| WebBytesSent | Number of retriever bytes sent |
| WebImageCacheHits | Number of successful hits on the image cache |
| WebImageCacheMiss | Number of misses on retriever image cache |
| WebActiveProcess | Number of active web retriever processes |
| WebBusyProcess | Number of busy web retriever processes |
| WebIdleProcess | Number of idle web retriever processes |
| WebTimeDuration | Elapsed time since retriever started |
| WebUrlFail | Number of retriever URL's that have failed |
| WebUrlRequested | Number of retriever URL's that have been requested |
| WebUrlSucceeded | Number of retriever URL's successfully accessed. |

Notes Database Stats

| | |
|------------------|---|
| BufCtlPoolPeak | Size of buffer control pool |
| BufCtlPoolUsed | Number of bytes used in buffer control pool |
| BufPoolPeak | Number of bytes allocated from buffer pool |
| BufPoolMax | Maximum size of buffer pool |
| BufPoolUsed | Number of used buffers from buffer pool |
| BufPoolPerCReads | Percentage of buffer pool reads |
| BufPoolReads | Number of database buffer pool reads |
| BufPoolWrites | Number of database buffer pool writes |
| NSFPoolPeak | Size of NSF pool |
| NSFPoolUsed | Amount of space used in NSF pool |
| NIFPoolPeak | Size of database NIF pool |
| NIFPoolUsed | Number of database NIF pools |

Notes Mail Stats

| | |
|-----------------|---|
| DeadMail | Number of dead (undeliverable) mail messages |
| DeliveredMail | Number of messages received by router |
| TotalMailFail | Total number of routing failures |
| TotalRoutedMail | Total number of mail messages routed |
| TransfMail | Number of messages router attempted to transfer |
| WaitingMail | Number of mail messages waiting to be routed |
| NumWaitingRecp | Number of pending mail messages awaiting local delivery |
| AvgMailDlvTime | Average time for mail delivery in seconds |
| AvgMailSrvHops | Average number of server hops for mail delivery |
| AvgMailSizeDlv | Average size of mail messages delivered in bytes |
| MaxMailDlvTime | Maximum time for mail delivery in second |
| MaxMailSrvHops | Maximum number of server hops for mail delivery |
| MaxMailSizeDlv | Maximum size of mail delivered in bytes |
| MinMailDlvTime | Minimum time for mail delivery in seconds |
| MinMailSrvHops | Minimum number of server hops for mail delivery |
| MinMailSizeDlv | Minimum size of mail delivered in bytes |
| TotalKBTransf | Total mail transferred in Kilobytes |
| MailTransfFail | Mail messages the router was unable to transfer |

Notes Server Stats

| | |
|------------------|---|
| Open.MaxUsers | Attempts to open MAXUSER server |
| Open.PreV4Client | Attempts to open server by pre V4 clients |

| | |
|---------------------|--|
| Open.V4Client | Attempts to open server by post V4 client |
| Open.Restricted | Attempts to open restricted server |
| Dropped_Sessions | Number of dropped sessions |
| TransPerMin | Average number of transactions per minute |
| TransPerMinPeak | Peak number of transactions n given minute |
| TransPerMinPeakT | Date and time for TransPerMinPeak |
| TransTotal | Total number of transactions the server has serviced |
| ServerUsers | Number of users with sessions open on the server |
| UsersPeak | Peak number of concurrent users with open sessions |
| UsersPeakTime | Date and time for UsersPeak |
| Domino Server Stats | |
| DomReqMin | Number of Domino requests in the last minute |
| DomReqMinPeak | Peak number of Domino requests per minute |
| DomReqMinPeakT | Time at one minute peak requests |
| DomReq5Min | Number of Domino requests in the last 5 minutes |
| DomReq5MinPeak | Peak number of Domino requests in 5 minutes |
| DomReq5MinPt | Time at one five minute peak requests |
| DomReqHour | Number of Domino requests in the last hour |
| DomReqHourPeak | Peak number of Domino requests in one hour |
| DomReqHourPeakT | Time at one hour peak requests |
| DomReqDay | Number of Domino requests in the last day |
| DomReqDayPeak | Peak number of Domino requests in one day |
| DomReqDayPeakT | Time at one day peak requests |
| DomReqTotal | Total Domino requests since the Domino server stats |
| Notes Memory Stats | |
| MemTotal | Total memory allocated |
| MemProcess | Total process-private memory allocated by ALL |
| MemShared | Total shared memory allocated |
| MemAvailability* | Availability of memory on this server |
| MemFree* | Free memory as reported by OS |
| MemSwapSize | Size of swap file |

For more information on the PTXNOTES Agent look at the IBM Intranet page <http://ncpdt.austin.ibm.com>. Customers should contact their IBM sales representative to get the program.

9.2 Analysis

The tools in 5.2, "Playback" on page 99, are useful for viewing relatively small recording files. If your recording files contain hours or days work of data, other methods of data review are needed. Additionally, viewing graphical performance data may not provide you with enough analysis of your performance problems.

9.2.1 Postprocessing Recording Files

Postprocessing recording files are used for the following:

1. Determine what data you want to collect.
2. Use one of the methods above to collect your data.
3. Split your recording files into files of like statistics.
4. Put the recorded data in tabular format.
5. Create spreadsheet macros to automate the creation of standard graphs.
6. Import you data to your spreadsheet.

Command files that are used for this process are:

- ptxls

- ptxtab
- ptxsplit
- ptxhottab
- ptx2stat

9.2.1.1 The ptxls Command

The ptxls command allows you to list the control record in a recording file. It can be used to determine if one or more control records exist in a recording file and to tell which statistics are contained in the recording file. Additionally, the time period of the recording file and the number of samples can be determined.

```
> ptxls /etc/perf/azizo.970723
# Configuration: ID=xmservd recording, Version=2.3
hosts/ah6100a/IP/NetIF/lo0/octet          00001/00001
hosts/ah6100a/IP/NetIF/lo0/opacket        00001/00002
hosts/ah6100a/IP/NetIF/lo0/ioctet         00001/00003
hosts/ah6100a/IP/NetIF/lo0/ipacket        00001/00004
hosts/ah6100a/Proc/runque                  00001/00005
hosts/ah6100a/Disk/hdisk0/wblk            00001/00006
hosts/ah6100a/Disk/hdisk0/rblk            00001/00007
hosts/ah6100a/Disk/hdisk0/busy            00001/00008
hosts/ah6100a/PagSp/totalfree             00001/00009
hosts/ah6100a/PagSp/totalsize             00001/00010
hosts/ah6100a/Mem/Virt/pgspgout           00001/00011
hosts/ah6100a/Mem/Virt/pgspgin           00001/00012
hosts/ah6100a/Mem/Virt/pageout            00001/00013
hosts/ah6100a/Mem/Virt/pagein            00001/00014
hosts/ah6100a/Mem/Real/numfrb             00001/00015
hosts/ah6100a/CPU/cpu0/wait               00001/00016
hosts/ah6100a/CPU/cpu0/kern               00001/00017
hosts/ah6100a/CPU/cpu0/user               00001/00018

# Statistics for above:  Start time Wed Jul 23 13:02:14 1997
#                               End time Thu Jul 24 16:58:31 1997
#                               Statset observation count      138
#                               Hotset observation count        0
# -----
```

Figure 116. Output from ptxls

Performance Toolbox includes some tools to convert your recording files to ASCII for later analysis.

9.2.1.2 The ptxtab Command

The ptxtab command lets you format a recording file for tabulated output. The program takes a recording file as input and produces one output file for each set of statistics (statset) in the recording file.

Each of the output files are named by suffixing the instrument sequence number to the name of the recording file. If the recording file has its original file name as created by xmperf of 3dmon, the initial R. is changed to A. to distinguish between Recording and ASCII output files. This also ensures that ASCII output files do not show up in the dialog window used to start playback.

The ptxtab command line is simple:

```
ptxtab [ -l lines | -c | -s ] recording_file_name
```

- l** This flag is used to specify the number of lines per page you want in the output file. The default is 23 lines per page. If you specify 0 (zero) lines per page, pagination is suppressed.
- c** This flag tells ptxtab to format the output in a comma-separated ASCII file suitable for postprocessing with programs such as awk, perl or SAS.
- s** This flag tells ptxtab to format the output suitable for importing into a spreadsheet program.

Note: The **-c**, **-l** and **-s** flags are mutually exclusive.

Refer to the *Performance Toolbox for AIX: Guide and Reference*, SC23-2625, Chapter 9 for more detailed information on all options with ptxtab.

```

"#Monitor: xmservd recording --- hostname: ah6100a"
"Timestamp" "CPU/cpu0/wait" "CPU/cpu0/kern" "CPU/cpu0/user"
"1997/07/23 13:02:14"      0.6      2.4      4.6
"1997/07/23 13:07:14"      0.2      0.6      0.4
"1997/07/23 13:12:05"      0.3      0.9      1.4
"1997/07/23 13:17:05"      0.2      0.6      0.5
"1997/07/23 13:22:05"      0.3      0.7      2.6
"1997/07/23 13:27:05"      0.2      0.7      1.2
"1997/07/23 13:32:05"      0.2      0.6      0.4
"1997/07/23 13:37:05"      2.2      1.0      1.3
"1997/07/23 13:42:05"      0.9      1.8      3.1
"1997/07/23 13:47:05"      0.3      0.8      1.1
"1997/07/23 13:52:07"      1.5      2.5      4.4
"1997/07/23 13:57:07"      0.3      1.8      3.7
"1997/07/23 14:02:07"      0.6      2.1      4.5

```

Figure 117. Output from ptxtab

9.2.1.3 The ptxsplit Command

Recording files usually contain many statistics that are unrelated to one another, at times even from different systems. To view the recording files with azizo or to perform postprocessing of the data, it is necessary to break the recording file down to smaller, more meaningful chunks. If the recording files contain more than one set of control records, splitting the file becomes a necessity.

There are numerous options that you may wish to use with the ptxsplit command. Below is a list of the more commonly used ones.

```
ptxsplit { -p parts | -s size | -h | -b | -f cfile | -d hmmm { -t dhhmm} infile
```

The command line arguments are all mutually exclusive, except that the **-t** argument is only valid if the **-d** argument is given. One of the arguments must be specified.

- h** Split into files according to host name of individual observations. The output file names are infile.hostname1, infile.hostname2, ..., infile.hostname*n*.
- b** Split into files for each set of control records encountered. The output file names are infile.b1, infile.b2, ..., infile.b*n*.

- f Split into two files. The flag must be followed by the file name of a control file. The first output file contains all of the statistics listed in the control file. The remaining statistics are written to the second output file. Statistics are specified in the control file with their full path name. If the host portion is omitted, statistics are selected across all host names. If the host portion is present, an exact match is required. The first file name is infile.sel; the second file is named infile.rem.

Refer to the *Performance Toolbox for AIX: Guide and Reference*, SC23-2625, Chapter 9 for more detailed information on all options with ptxsplit.

9.2.1.4 The ptxhottab Command

The ptxhottab program is a new program, included after support for HotSet data was added to PTX. The program is used to tabulate data from a recording file like ptxtab, but it tabulates HotSet data while ignoring other data.

The ptxhottab command line format:

ptxhottab [-c] recording_file

- c Condensed output

The program has two output formats: uncondensed and condensed. Below are samples of both. Uncondensed output uses a format with name=equals-value pairs separated by semicolons. It looks like this (path names truncated):

```
date=1997/08/04;time=08:42:24;value=85.051;threshold=35;path=hosts/bleikamp/Disk/hdisk0/busy
date=1997/08/04;time=10:55:44;value=71.699;threshold=35;path=hosts/bleikamp/Disk/hdisk1/busy
date=1997/08/04;time=10:55:44;value=58.499;threshold=35;path=hosts/bleikamp/Disk/hdisk0/busy
date=1997/08/04;time=10:56:05;value=40.097;threshold=35;path=hosts/bleikamp/Disk/hdisk1/busy
date=1997/08/04;time=13:37:23;value=58.500;threshold=35;path=hosts/bleikamp/Disk/hdisk0/busy
date=1997/08/04;time=13:37:33;value=54.300;threshold=35;path=hosts/bleikamp/Disk/hdisk0/busy
date=1997/08/04;time=13:37:53;value=55.996;threshold=35;path=hosts/bleikamp/Disk/hdisk0/busy
date=1997/08/04;time=13:38:33;value=35.000;threshold=35;path=hosts/bleikamp/Disk/hdisk0/busy
date=1997/08/04;time=13:38:43;value=58.000;threshold=35;path=hosts/bleikamp/Disk/hdisk0/busy
date=1997/08/04;time=14:14:53;value=39.500;threshold=35;path=hosts/bleikamp/Disk/hdisk1/busy
date=1997/08/04;time=14:15:43;value=59.445;threshold=35;path=hosts/bleikamp/Disk/hdisk1/busy
date=1997/08/04;time=14:15:43;value=37.965;threshold=35;path=hosts/bleikamp/Disk/hdisk0/busy
date=1997/08/04;time=15:59:04;value=52.387;threshold=35;path=hosts/bleikamp/Disk/hdisk0/busy
date=1997/08/04;time=15:59:04;value=47.977;threshold=35;path=hosts/bleikamp/Disk/hdisk1/busy
```

Figure 118. Uncondensed Output from ptxhottab

```

970804 08:42      85.051    35 hosts/bleikamp/Disk/hdisk0/busy
970804 10:55      71.699    35 hosts/bleikamp/Disk/hdisk1/busy
970804 10:55      58.499    35 hosts/bleikamp/Disk/hdisk0/busy
970804 10:56      40.097    35 hosts/bleikamp/Disk/hdisk1/busy
970804 13:37      58.500    35 hosts/bleikamp/Disk/hdisk0/busy
970804 13:37      54.300    35 hosts/bleikamp/Disk/hdisk0/busy
970804 13:37      55.996    35 hosts/bleikamp/Disk/hdisk0/busy
970804 13:38      35.000    35 hosts/bleikamp/Disk/hdisk0/busy
970804 13:38      58.000    35 hosts/bleikamp/Disk/hdisk0/busy
970804 14:14      39.500    35 hosts/bleikamp/Disk/hdisk1/busy
970804 14:15      59.445    35 hosts/bleikamp/Disk/hdisk1/busy
970804 14:15      37.965    35 hosts/bleikamp/Disk/hdisk0/busy
970804 15:59      52.387    35 hosts/bleikamp/Disk/hdisk0/busy
970804 15:59      47.977    35 hosts/bleikamp/Disk/hdisk1/busy

```

Figure 119. Condensed Output from `ptxhottab`

9.2.1.5 The `ptx2stat` Command

The `ptx2stat` program is a new program, included after support for HotSet data was added to PTX. The program is used to modify the HotSet data collected in a recording file so it appears to be collected in StatSet(s). Because HotSets and StatSets are inherently different, and especially because HotSet data is likely to exist for only a few time periods, the conversion can normally not make the data appear exactly as StatSet data.

The erratic scattering of actual observations makes it difficult to see observations over a time period when played back with `xmperf`. To make it more usable, it is a good idea to postprocess the converted recording file so that the old HotSet data appears in the same StatSet as real StatSet data. The converted HotSet observations then appear as “blips” in the playback window.

For use with `azizo`, conversion with `ptx2stat` can be done so each observation is followed by a stop record.

The `ptx2stat` command line format:

```
ptx2stat [-s] infile outfile
```

-s Insert stop-records after each value set.

```

# Configuration: ID=xmservd recording, Version=2.3
hosts/bleikamp/Disk/*/busy           00003
hosts/bleikamp/Disk/hdisk1/busy      00001/00001
hosts/bleikamp/Disk/hdisk0/busy      00001/00002

# Statistics for above:  Start time Mon Aug  4 08:42:24 1997
#                        End time Mon Aug  4 16:57:43 1997
#                        Statset observation count      87
#                        Hotset observation count        11
# -----

```

Figure 120. Listing of Recording File before `ptx2stat`

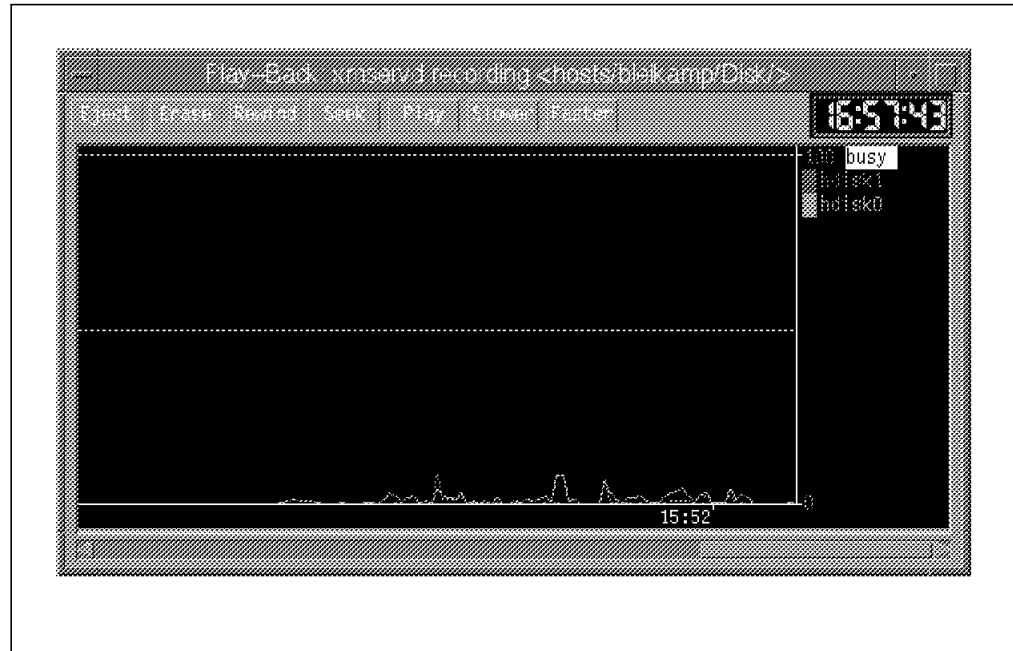


Figure 121. The xperf Playback before ptx2stat

Executing ptx2stat converted the HotSet observations to StatSet observations:

```
.# Configuration: ID=xmservd recording, Version=2.3
hosts/bleikamp/Disk/hdisk1/busy          00003/00001
hosts/bleikamp/Disk/hdisk0/busy          00003/00002
hosts/bleikamp/Disk/hdisk1/busy          00001/00001
hosts/bleikamp/Disk/hdisk0/busy          00001/00002

# Statistics for above:  Start time Mon Aug  4 08:42:24 1997
#                       End time Mon Aug  4 16:57:43 1997
#                       Statset observation count      98
#                       Hotset observation count        0
# -----
```

Figure 122. Listing of Recording File after ptx2stat

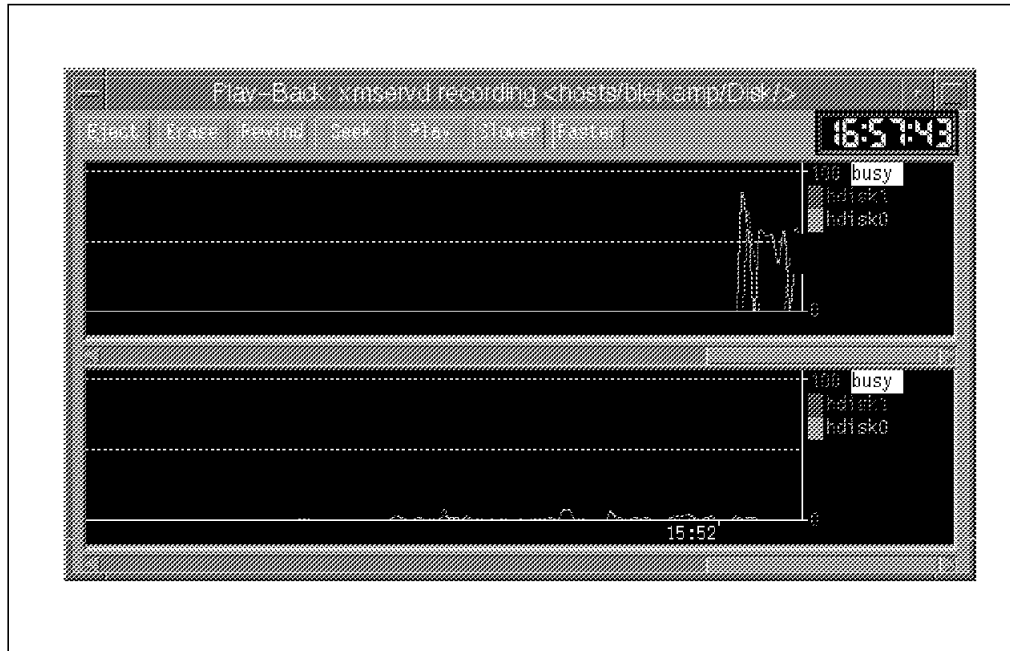


Figure 123. The xmperv Playback after ptx2stat

The HotStats that exceeded the specified thresholds in the recording files are now clearly visible in the playback window. The upper instrument is the playback of the HotStat; the lower instrument is the playback of the StatSet.

9.2.2 Script to Postprocess Recording Files

See A.2, "Sample Script to Split Recording Files" on page 178, for the complete text of the scripts.

9.2.3 Spreadsheets

By now, you may have decided that the graphs output by azizo may not be exactly what you want to use to track system performance. You may prefer to take the output of ptxtab and import the data to into a spreadsheet. Since there are numerous spreadsheets in use today, we do not attempt to detail graph creation.

Appendix A. Sample Scripts

This Appendix contains two sample scripts that we found useful.

A.1 Sample Script to Create a Customized xmservd.cf File

Script to build xmservd.cf with typical recording statistics:

```
#!/usr/bin/ksh
#set -x
#
# Script to build xmservd.cf with typical recording statistics
#
#

usage ()
{
echo "USAGE:\t$0 [ -c | -f ]"
echo "$0 will build an xmservd recording file"
echo "\t-c check configuration file with xmscheck"
echo "\t-f forces defaults to be used - no prompts"
echo "\t\toutput_file=./xmservd.cf"
echo "\t\tretain 14 days of data"
echo "\t\t1 day of day per file"
echo "\t\tsample every 5 minutes"
echo "\t\trecord Mon-Fri"
echo "\t\trecord 7am-7pm"
echo
exit 2
}

main ()
{
while getopts :cf flag
do
case $flag in
c)check="true" ;;
f)force="true" ;;
\?)echo "\n$0: unknown option $OPTARG"
usage $* ;;
esac
done
shift OPTIND-1

set_defaults

if [ -z "$force" ]
then
get_values
fi

check_values

build_config

if [ -n "$check" ]
then
```

```

check_file
fi
}

root_check ()
{
if [ `whoami` != "root" ]
then
echo "\nYou must be root to execute xmscheck."
echo "Checking of $CFGFILE not performed.\n"
exit
fi
}

set_defaults ()
{
CFGFILE_dflt=xmservd.cf
RETAIN_dflt=14
PERFILE_dflt=1
MININT_dflt=5
FREQUENCY_dflt=300000
STARThh_dflt=7
STARTdd_dflt=1-5
STARTmm_dflt=0
ENDhh_dflt=19
ENDdd_dflt=1-5
ENDmm_dflt=0
}

get_values ()
{
echo "\nEnter configuration file name (default=xmservd.cf) \c"
read CFGFILE
echo "\nHow many days of data do you wish to retain? (default=14) \c"
read RETAIN
echo "\nHow many days of data per file? (default=1) \c"
read PERFILE
echo "\nHow many minutes between recording samples? (default=5) \c"
read MININT
echo "\nWhich day range do you wish to use?"
PS3=' Please enter a number: '
select i in Sunday-Saturday Monday-Friday Other
do
case $REPLY in
1)STARTdd=1-5 ; ENDdd=1-5 ;;
2)STARTdd=0-6 ; ENDdd=0-6 ;;
3)echo "Enter day(s) to start recording, comma separated (0-Sun,6=Sat) \c"
read STARTdd
echo "Enter day(s) to end recording, comma separated (0-Sun,6=Sat) \c"
read ENDdd ;;
esac
break
done
echo "\nEnter hour(s) to start recording (24 hour clock default=7) \c"
read STARThh
echo "\nEnter minute(s) to start recording (default=0) \c"
read STARTmm
echo "\nEnter hour(s) to end recording (24 hour clock default=19) \c"
read ENDhh
}

```



```
echo "\nEnter minute(s) to end recording (default=0) \c"
read ENDmm
}

check_values ()
{
if [ -z "$CFGFILE" ]
then
CFGFILE=$CFGFILE_dflt
fi
if [ -f "$CFGFILE" ]
then
datetag=`date "+%m%d%y" `
mv $CFGFILE ${CFGFILE}.${datetag}
echo "Previous version of $CFGFILE was moved to ${CFGFILE}.${datetag}"
fi
if [ -z "$RETAIN" ]
then
RETAIN=$RETAIN_dflt
fi
if [ -z "$PERFILE" ] || [ "$PERFILE" -gt "$RETAIN" ]
then
PERFILE=$PERFILE_dflt
fi
if [ -z "$MININT" ]
then
MININT=$MININT_dflt
fi
let FREQUENCY=$MININT*60000
if [ -z "$STARTdd" ]
then
STARTdd=$STARTdd_dflt
fi
if [ -z "$STARThh" ]
then
STARThh=$STARThh_dflt
fi
if [ -z "$STARTmm" ]
then
STARTmm=$STARTmm_dflt
fi
if [ -z "$ENDdd" ]
then
ENDdd=$ENDdd_dflt
fi
if [ -z "$ENDhh" ]
then
ENDhh=$ENDhh_dflt
fi
if [ -z "$ENDmm" ]
then
ENDmm=$ENDmm_dflt
fi
}

build_config ()
{
echo "#" > $CFGFILE
echo "# XMSERVD Recording configuration file">> $CFGFILE
}
```

```

echo "#">> $CFGFILE
echo "# Keep files for at least x days ">> $CFGFILE
echo "# let each file contain y days recordings">> $CFGFILE
echo "# retain x y">> $CFGFILE
echo "retain $RETAIN $PERFILE">> $CFGFILE
echo "">> $CFGFILE
echo "# Set default sampling interval (1 minute=60000)">> $CFGFILE
echo "# frequency 60000">> $CFGFILE
echo "frequency $FREQUENCY">> $CFGFILE
echo "">> $CFGFILE
echo "# List stats to record at the default frequency">> $CFGFILE
echo "CPU/cpu0/user">> $CFGFILE
echo "CPU/cpu0/kern">> $CFGFILE
echo "CPU/cpu0/wait">> $CFGFILE
echo "">> $CFGFILE
echo "Mem/Real/numfrb">> $CFGFILE
echo "Mem/Virt/pagein">> $CFGFILE
echo "Mem/Virt/pageout">> $CFGFILE
echo "Mem/Virt/pgspgin">> $CFGFILE
echo "Mem/Virt/pgspgout">> $CFGFILE
echo "">> $CFGFILE
echo "PagSp/totalsize">> $CFGFILE
echo "PagSp/totalfree">> $CFGFILE
echo "">> $CFGFILE
for i in `lsdev -Cc disk | cut -f1 -d" "`
do
echo "Disk/${i}/busy">> $CFGFILE
echo "Disk/${i}/rblk">> $CFGFILE
echo "Disk/${i}/wblk">> $CFGFILE
done
echo "">> $CFGFILE
echo "Proc/runque">> $CFGFILE
echo "">> $CFGFILE
for i in `lsdev -Cc if -r name -S available`
do
echo "IP/NetIF/${i}/ipacket">> $CFGFILE
echo "IP/NetIF/${i}/ioctet">> $CFGFILE
echo "IP/NetIF/${i}/opacket">> $CFGFILE
echo "IP/NetIF/${i}/oocet">> $CFGFILE
done
echo "">> $CFGFILE
echo "# More stats to record and another frequency">> $CFGFILE
echo "# value frequency">> $CFGFILE
echo "">> $CFGFILE
echo "# recording times ">> $CFGFILE
echo "start $STARTdd $STARThh $STARTmm $ENDdd $ENDhh $ENDmm">> $CFGFILE
}

check_file ()
{
root_check
xmscheck $CFGFILE > xmscheck.out
echo "\nOutput from xmscheck is xmscheck.out"
grep "RECORDING IS NOT ACTIVE" xmscheck.out >/dev/null
if [ $? -eq 0 ]
then
echo "One or more errors exist in $CFGFILE\n"
else
echo "\n$CFGFILE is valid"

```

```
if [ `pwd` != "/etc/perf" ]
then
echo "Move $CFGFILE to /etc/perf and refresh xmservd to activate\n"
else
echo "Refresh xmservd to activate\n"
fi
fi
}

main $*
```

A.2 Sample Script to Split Recording Files

Script to split and tabulate recording file data:

```
#!/usr/bin/ksh
#
#Script to split and tabulate recording file data
#$1 is the name to the recording file
#
function usage
{
print "\nUSAGE: $0 [ -s ] [ -t ] recording_file "
print " -s Only split recording files"
print " -t Only execute ptxtab on all files"
exit 2
}
#
TAB=true
SPLIT=true

while getopts :st flag
do
    case $flag in
    s) TAB=false ;;
    t) SPLIT=false ;;
    \?) print "$0: unknown option $OPTARG"
        usage ;;
    esac
done
shift OPTIND-1

recording=`basename $1`
if [ ${recording%%.*} = "azizo" ]
then
splitdir=/etc/perf/ptxsplit.dir
AZIZO=true
else
splitdir=$HOME/XmRec/ptxsplit.dir
fi
echo "\nOutput files will be placed in ${splitdir}\n"

if [ $SPLIT = "true" ]
then
if [ $# -ne 1 ]
then
echo "\n==> Name of recording file must be specified on invocation."
usage
exit 1
fi

if [ "$recording" = $1 ] && [ `pwd` = $splitdir ]
then
echo "=> Recording file $1 will be modified."
else
if [ ! -d $splitdir ]
then
mkdir $splitdir
fi
cd $1 $splitdir
```

```

cd $splitdir
fi

ptxls $1 | awk '
# awk script to split out different record types from a recording file
#
BEGIN {
    FS = "/"
    type[1] = "CPU"
    type[2] = "Mem"
    type[3] = "PagSp"
    type[4] = "Disk"
    type[5] = "LAN"
    type[6] = "Proc"
    type[7] = "Syscall"
    type[8] = "SysIO"
    type[9] = "IPC"
    type[10] = "FS"
    type[11] = "IP"
    type[12] = "TCP"
    type[13] = "UDP"
    type[14] = "RPC"
    type[15] = "NFS"
    type[16] = "Spmi"
    type[17] = "DDS"
    maxtypes = 17
}
/##/ { next }
{
    for ( i = 1 ; i <= maxtypes ; i++ ) {
        if ( $3 == type[i] ) {
            print $0 >> "split."type[i]
        }
    }
}
}'
#
hostlist=`ptxls $recording | grep "hosts/" | cut -f2 -d/ | sort | uniq`

ptxsplit -h $1

for host in $hostlist
do
print
for i in split.*
do
type=${i##split.}
echo "=> Executing ptxsplit Hostname=$host - Record Type=$type ... "
ptxsplit -f $i ${recording}.${host}
mv ${recording}.${host}.sel ${recording}.${host}.${type}
mv ${recording}.${host}.rem ${recording}.${host}
#rm $i
done
mv ${recording}.${host} ${recording}.${host}.other
numother=`ptxls ${recording}.${host}.other | grep hosts | wc -l`
if [ $numother -eq 0 ]
then
rm -f ${recording}.${host}.other
fi
done

```

```
rm split.*
fi
#
if [ $TAB = "true" ]
then
cd $splitdir
if [ "$AZIZO" = "true" ]
then
filelist="{recording}.*.*"
else
filelist="{recording}.*.*"
fi

if [ ! -f ${filelist} ]
then
echo "\n==> No recording files to tabulate."
else
print
for i in ${filelist}
do
if [ $i = ${i%}_"* ]
then
echo "====> Executing ptxtab for $i ..."
ptxtab -s $i
fi
done
fi
fi
```

A.3 Regular Data Gathering with vmstat and Time Stamps

Here are some scripts to help with the data collection of vmstat. The nice feature about this is that the final result does have a time stamp in front of each vmstat output line.

A.3.1 The vmit script

This script is called vmit. This script adds a date/time stamp to vmstat output. Execute it using for example:

```
vmit 60 600 | tee /tmp/vmstat.(date)
```

A daily cron entry would be useful for the vmit command.

```
#!/bin/ksh
# vmit
# This script adds a Date/Time to vmstat output so it is
# easy to see when the system was stressed.

if ( test $# -ne 1 && test $# -ne 2 ) then
  echo "usage: $0 <seconds between vmstat info > [<time interval>]"
  exit 1
fi

export interval=$1
if ( test $# -eq 2 ) then
  export times=`expr $2 + 1`
else
  export times=""
fi

echo "Interval: " $interval "Date/Time:" `date`
vmstat $interval $times

exit 0
```

A.3.2 The format_vm Script

The format_vm script is for formatting the output file from the vmit script:

```
format_vm /tmp/vmstat.(date) > /tmp/output_vm_file
```

Here it is:

```
#!/bin/ksh
# format_vm
# This script formats the output from the vmstat command so the
# proper time is at the beginning of each line

if ( test $# -ne 0 -a $# -ne 1 ) then
  echo "usage: $0 <file name (def stdin)>"
  exit 1
fi

if ( test $# -eq 0 ) then
  export file=""
else
  export file=$1
fi
```

```

awk 'BEGIN {
    initialize = 0
}
initialize == 1 {
    if ( $1 == "kthr" ) {
# skip this line and the next two lines
        getline
        getline
    } else {
# put time stamp before vmstat info
        printf "%.2d:%.2d:%.2d %s\n", hour, min, sec, $0

# update sec, min, hour
        sec += delta
        if ( sec >= 60 ) {
            min += 1
            sec -= 60
        }
        if (min >= 60 ) {
            hour += 1
            min -= 60
        }
        if ( hour > 23 ) {
            hour -= 24
        }
    }
}
initialize == 0 {
    initialize = 1
    delta = $2
    split($7, A, ":")
    hour = A[1]
    min = A[2]
    sec = A[3]
    print $0
# skip the initial header and the summary line
    getline
    printf "          %s\n", $0
    getline
    printf " TIME   %s\n", $0
    getline
    printf "          %s\n", $0
    getline
}' $file

```

Appendix B. LMON: Developing a Local Monitor

The `lmon` program is a sample application we developed from the PTX sample program `/usr/samples/perfagent/server/lchmon.c`. The complete program is included in source form on the accompanying media. In the text, we discuss the changes we made to the original program to develop `lmon`. The `lmon` program differs from `lchmon` in that it can also generate an ASCII output file suitable for input to `a2ptx`.

We chose `lchmon` as the basis for this example because it uses the `Spmi` API. Any node with the LPP `perfagent.server` installed can use this program.

Just remember that with the Performance Toolbox architecture there are three APIs available, one for each LPP. To see the similarities between the `Spmi` and `Rsi` APIs, compare the programs `/usr/samples/perfagent/server/lchmon.c` and `/usr/samples/perfmgr/chmon.c`. The structure of the two programs is nearly identical. The major change is the library accessed. Reviewing, we get:

| | |
|---------------------|---|
| perfagent | <code>Spmi</code> (System Performance Measurement Interface) |
| perfmgr | <code>Rsi</code> (Remote Statistics Interface) |
| ptpe.program | <code>PTPE</code> (Performance Toolbox Parallel Extensions). Note that samples for <code>PTPE</code> may not be located in <code>/usr/samples/ptpe/</code> . If you can't find them, look in <code>/usr/lpp/ptpe/samples</code> . |

When you look at the original program, there are so many `#ifdefs` blocks that readability is impaired. You may want to strip them out before working on an application of your own. The sequence in Figure 124 can be used to simplify the source.

```
sed -e '/-#include/d' <lchmon.c >lmon.c
cc -P -D_AIX lmon.c
egrep '-#include' <lchmon.c >lmon.c
sed -e '/-$/d' -e '/-$/d' <lmon.i >>lmon.c
rm lmon.i
```

Figure 124. Sample Program to Simplify `lchmon.c` Code

The rest of this Appendix is organized as follows:

- Discussion of `lchmon` data structures and program flow
- Discussion of the modifications made to develop `lmon`
- Hints & tips for using `lmon` or another program similar to it

B.1 Key Data Structures in `lchmon`

In this section, the line numbers printed with the source code are from the program source, `lchmon.c`. We did not include line numbers on the additions to ease comparison to the original program as you study this Appendix.

There are several global symbols defined in `lchmon`:

- `legend`
- `val1`
- `prt1`

- ssp
- svp

The program is easy to understand when you see it using these structures. The first two structures are used to define the fixed text we see on the screen, `legend[]`, (see Figure 125) and names of the statistics we want to monitor, `val1[]` (see Figure 126). The last static array is `prt1[]` (see Figure 127 on page 185). This array, defined as `struct` where, is used within a loop to print the `StatSet` defined from `val1[]`. We discuss `ssp` and `svp` below (see Figure 128 on page 186.)

```

274 char *legend[] = {
275 "Spmi Data User API      Local Monitor for host",
276 "LCHMON Sample Program  ***      ***      Interval:      seconds",
277 "" ,
278 #ifdef _AIX
279 "% CPU                      EVENTS/QUEUES  FILE/TTY",
280 "Kernel                    | Pswitch      Readch",
281 "User                      | Syscall      Writech",

```

Figure 125. An Excerpt of the Variable `legend[]`

Figure 125 shows how the fixed part of the screen is built up. At line 278, there is a beginning of an `#ifdef` block which is specific for AIX statistic naming conventions.

```

354 char *val1[] = {
355 "CPU/glkern",
356 "CPU/gluser",
357 #ifdef _INCLUDE_HPUX_SOURCE
358 "CPU/glnice",
359 #endif /* _INCLUDE_HPUX_SOURCE */
360 #ifdef _SunOs
361 "CPU/glnice",
362 #else
363 "CPU/glwait",
364 #endif /* _SunOs */
365 "CPU/glidle",
366 "Mem/Virt/pagexct",
367 #ifdef _SOLARIS
368 "Mem/Virt/zerofill",
369 #else /* _SOLARIS */
370 "Mem/Virt/steal",
371 #endif /* _SOLARIS */
372 "Mem/Virt/pgrc1m",
373 #if defined(_AIX) || defined(_SOLARIS)
374 "Mem/Virt/pgpggin",
375 "Mem/Virt/pgspgout",

```

Figure 126. An Excerpt of the Definition of the Variable `val1[]`

Figure 126 shows how strings are defined in the array `val1[]`. These strings are PTX names of statistics that will be collected. Note the platform-specific naming convention for some of the statistics.

```
91 typedef struct
92 {
93     int    line;
94     int    col;
95 #ifdef  _NO_PROTO
96     void   (*fun)();
97 #else
98     void   (*fun)(long, int, int);
99 #endif
100     float  mul;
101 } where;

450 where prt1[] = {
451 #ifdef  _INCLUDE_HPUX_SOURCE
452 {3, 10, p31, 10.0},
453 {4, 10, p31, 10.0},
454 {5, 10, p31, 10.0},
455 {6, 10, p31, 10.0},
456 {7, 10, p31, 10.0},
457 #else
458 {4, 8, p31, 10.0},
459 {5, 8, p31, 10.0},
460 {6, 8, p31, 10.0},
461 {7, 8, p31, 10.0},
462 #endif /* _INCLUDE_HPUX_SOURCE */
463 {10, 8, p5, 1.0},
```

Figure 127. Definition of prt1 with the Structure where

Figure 127 shows how the last coded symbol is initialized in the source. The structure where (starting at line 91) defines the layout of prt[]. Again, from this excerpt, you can see that the actual data defining the layout of the displayed text is dependent on the target system.

B.2 Program Flow in lchmon

The program flow is simple to describe in terms of:

- Initialization
- Collection
- Display
- Termination

B.2.1 Initialization

The program lchmon begins by checking the command line options given, setting up some signal() catching, and setting up the static text on the screen. After this is finished and SpmiInit() has been called successfully, the routine lststat() is called. This routine uses val1[] to initialize svp.

The subroutine lststat() first creates an empty Spmi SpmiStatSet context and an empty (for instance, no SpmiStats have been added) SpmiStatVals data structure, or *container*. During program execution, the context and container are passed to Spmi, which fills the container's SpmiStats with current values. The returned container (reference) is filled with statistics all obtained simultaneously. When Spmi is called to report statistics, all of the Spmi StatVals structure in the SpmiStatSet are updated. Hereafter, we refer to these structures collectively as a *StatSet*.

Review: The basic steps needed are:

1. Call `SpmiInit()` to set up communications with the `perfagent.server` kernel extension.
2. Establish the `StatSet`.

```

261 struct SpmiStatSet    *ssp;
262 struct SpmiStatVals  *svp[512];

782     if (!(ssp = SpmiCreateStatSet()))
783     {
784         fprintf(stderr, "SpmiCreateStatSet can't create StatSet\n");
785         begone();
786         exit(62);
787     }
788
789     for (m = 0; m < tabsize; m++)
790     {
791         strcpy(tmp, basepath);
792         strcat(tmp, val1[m]);
793         svp[m] = SpmiPathAddSetStat(ssp, tmp, NULL);
794     }

```

Figure 128. The Basic Sequence Used to Create and Define a `StatSet`

Beside the base statistics specified in `val1[]`, there are additional statistics that `lchmon` includes in the `StatSet`. The number of these variables is determined at runtime (the Disk and IP/NetIF or LAN statistics). Figure 129 shows how data statistics can be specified and added dynamically.

```

796     strcpy(tmp, basepath);
797     strcat(tmp, "Disk");
798     if ((i = addcont(tabsize, 2, tmp)) == -1)
799     {
800         if (strlen(SpmiErrmsg))
801             fprintf(stderr, "%s", SpmiErrmsg);
802         begone();
803         exit(65);
804     }
805     ix2 = i - m;
806     m = i;
807
808     strcpy(tmp, basepath);
809 #ifdef _SOLARIS
810     strcat(tmp, "LAN");
811 #else /* _SOLARIS */
812     strcat(tmp, "IP/NetIF");
813 #endif /* _SOLARIS */
814     if ((i = addcont(m, 3, tmp)) == -1)
815     {
816         if (strlen(SpmiErrmsg))
817             fprintf(stderr, "%s", SpmiErrmsg);
818         begone();
819         exit(65);
820     }

```

Figure 129. How Statistics Can be Added Dynamically

Figure 130 on page 187 shows more detail on how you can add statistics dynamically. Note that only the code needed for actually updating the `StatSet` is shown. The missing lines are used to initialize the screen display.

```

675  int addcont(int ix, int six, char *path)
677  {
678      int    i = ix, n = 0;
679      char  tmp[128];
680      char  *s;
681      SpmiCxHdl  cxh;
682      struct SpmiCx  *cxp;
683      struct SpmiStatLink  *statlink;
684      struct SpmiCxLink  *cxlink;
685
686      cxh = SpmiPathGetCx(path, NULL);
687      if (!cxh)
688      {
689          if (strlen(SpmiErrMsg))
690              fprintf(stderr, "%s", SpmiErrMsg);
691          fprintf(stderr, "SpmiPathGetCx can't access path %s\n",
692                  path);
693          return(i);
694      }
695
696      if ((cxlink = SpmiFirstCx(cxh)))
697      {
698          while (cxlink)
699          {
700              if (!(cxp = SpmiGetCx(cxlink->context)))
701                  break;
702
703              strcpy(tmp, path);
704              if (strlen(tmp))
705                  strcat(tmp, "/");
706              if (cxp->name)
707                  strcat(tmp, cxp->name);
708              if ((i = addstat(i, six, tmp)) == -1)
709              {
710                  if (strlen(SpmiErrMsg))
711                      fprintf(stderr, "%s", SpmiErrMsg);
712                  begone();
713                  exit(63);
714              }
715              cxlink = SpmiNextCx(cxlink);
716          }
717      }
718      return(i);
719  }

```

Figure 130. More Detail for Adding Statistics Dynamically

B.2.2 Collection

Once the StatSet has been established, collection and reporting of data can begin. The lchmon program uses `setitimer()` to set up the collection interval, `pause()` to wait for the interval timer, and `signal(SIGALRM, feeding)` is used to direct processing to the procedure `feeding()` at each interval. See the code from `main()` in Figure 131 on page 188. The actual `feeding()` code is in Figure 137 on page 192.

```

1081  /*
1082      Do actual processing
1083  */
1084  apath[0] = '\0';
1085  lststats(apath);
1086  t_value.it_interval.tv_sec = interval/1000;
1087  if (t_value.it_interval.tv_sec < 1)
1088      t_value.it_interval.tv_sec = 1;
1089  t_value.it_interval.tv_usec = 0;
1090  t_value.it_value.tv_sec = t_value.it_interval.tv_sec;
1091  t_value.it_value.tv_usec = 0;
1092  if (setitimer(ITIMER_REAL, &t_value, &o_value) < 0)
1093  {
1094      begone();
1095      printf("error no %d while setting interval timer\n", errno);
1096      must_exit();
1097  }
1098  #ifdef _NO_PROTO
1099  signal(SIGALRM, (void(*)())feeding);
1100  #else
1101  signal(SIGALRM, (void(*) (int))feeding);
1102  #endif /* _NO_PROTO */
1103
1104  while (TRUE)
1105  {
1106      pause();
1107  }
1108

```

Figure 131. Sequence to Initiate and Maintain the Timed Collection of Data

B.2.3 Termination

We haven't found anything in the documentation to indicate that an unexpected exit from `Spmi` will leave the system in a unknown state, but it is better to call `SpmiExit()` when collection is to terminate. There may be other reasons why a program may want to have a chance to clean up before actually exiting. For example, in `lchmon` the terminal settings must be reset to their original state.

The easiest way to exit `lchmon` or `lmon` is to press **q** or **Q** (quit). At the next timed interval, the program will call its termination routines.

Review: The two most important environmental settings that need to be reset by `lchmon` are resetting the terminal mode from raw/cbreak mode to echo mode, and of course, giving the `Spmi` API a chance to clean up its shared memory structures, which it does by calling `SpmiExit()`. We didn't modify any of this code.

B.3 Making lmon

Remember, our goal is to have ASCII output suitable for input for `a2ptx`. After running `a2ptx`, we can use `xmperf` and/or `azizo` to analyze the data (see 5.2, "Playback" on page 99, and 9.2, "Analysis" on page 166). Only some code to generate data in a proper format was needed. We also wanted to maintain the original cursor's screen output. The cosmetic changes were done in order to parse for the name of an output file.

```
lmon [-a <filename>] [-i interval_seconds] [-p processes]
```

First, we had to get to get the ASCII file initialized with the proper heading information and thereafter, fill it with the data.

B.3.1 Input Format for a2ptx

See Figure 132 for an excerpt of the output generated by lmon. The four components are described below.

- Host identifier** This is the token *Hostname:* followed by a value that will prefix all the statistical names during the display by xmperf and azizo. This is absent or found as the last text on the first line.
- Statistic names** These are the names of the statistics, on a column by column basis, that are in the text file. These are specified on the second line.
- Time stamps** This is always the first statistic of every line. It consists of two parts separated by white space:
YYYY/MM/DD hh:mm:ss
YYYY is year; MM is month; DD is day of month; hh is hour; mm is minute; and ss is second. Note that with the white space between the two parts of this statistic, the data rows will always seem to have one more column than statistic names.
- Data values** The data values start directly after the statistic names and continue to the end of the file. The first statistic is always the timestamp in the layout specified above. Every statistic must have a value. Values may include a decimal point. If there is no value a dash (-) must be placed instead. That is, the number of data values must equal the number of statistic names (column headings).

```
Hostname: ah6100a
Timestamp CPU/glkern CPU/gluser CPU/glwait Disk/hdisk0/busy ...
1997/08/29 20:15:06 0.66 0.00 0.00 0.00 ...
1997/08/29 20:15:09 1.00 0.00 0.67 6.67 ...
1997/08/29 20:15:12 0.67 0.00 0.00 0.00 ...
```

Figure 132. Sample of a2ptx Input File

B.3.2 Initialization

Adding the hostname was trivial. We used the definition of a *FILE* pointer, a2ptx_out, to determine whether the ASCII output needed to be generated or not. See Figure 133.

```
1081  /*
1082      Do actual processing
1083  */
      #ifdef MAMFELT
      /*
      * Note: Hostname: token must be separated by white space from
      * the string value following
      * The next line starts the definition of the statistic names.
      * The first statistic is always Timestamp (or "Timestamp" )
      */
          if (a2ptx_out != NULL)
              fprintf(a2ptx_out, "Hostname: %s\nTimestamp", host);
      #endif
1084  apath[0] = '\0';
1085  lststats(apath);
```

Figure 133. Code Initializing lmon Output: Hostname and Timestamp

Creating the next output line required some more thought. However, once the ideal location(s) had been identified, the change was easy. **Note:** This addition is needed before each `SpmiPathAddSetStat()` call. We show only one addition in Figure 134 on page 190.

```

789         for (m = 0; m < tabsize; m++)
790         {
791             strcpy(tmp, basepath);
792             strcat(tmp, vall[m]);
           #ifdef MAMFELT
               if (a2ptx_out != NULL) /* Header Information */
                   fprintf(a2ptx_out, "%s", tmp);
           #endif
793             svp[m] = SpmiPathAddSetStat(ssp, tmp, NULL);
794         }

```

Figure 134. Code to Print the Names of the Statistics we are Collecting

Remember that your first statistic in your ASCII recording file is the timestamp. See the text starting lines 3-5 in Figure 132 on page 189. Since it has a space character (ASCII 32) as part of its format, a simple awk program to check the number of columns should show a difference of one (1) in the number of columns (or number of fields - NF - in awk). The simple program (Figure 135) helped check for errors.

```

$ awk ' { print NF } ' < my_a2ptx_input | uniq
2
42
43
$

```

Figure 135. Simple Program to Verify an a2ptx Input

B.3.3 Collection

Obviously, the collection of the data was already being accomplished. All that needed to be done was to get these values to the output stream. We made a separate print function so that we could take care of any possible errors caused by a missing or an illegal value.

Note

In the *Performance Toolbox for AIX: Guide and Reference, SC23-2526, Chapter 9*, it is stated: *Data values may have a decimal point or be integers. Missing values should be represented by a single dash (-).*


```

843 void feeding()
844 {
845     int          i, m, l, c;
846     float        f;
847     long         v;
848
849     signal(SIGALRM, SIG_IGN);
850     gettimeofday(&tv, NULL);
851     move(0,55);
852     addstr(ctime((time_t *)&tv));
853
854     if (SpmiGetStatSet(ssp, TRUE))
855     {
856         if (SpmiErrno == SiLocked)
857         {
858 #if defined(_AIX) || defined(_SunOs) || defined(_SOLARIS)
859             if (should_quit())
860 #else
861             if (((c = getch()) == 'q') || (c == 'Q'))
862 #endif /* _AIX or _SunOs or _SOLARIS */
863             {
864                 begone();
865                 must_exit();
866             }
867 #ifdef _NO_PROTO
868             signal(SIGALRM, (void(*)())feeding);
869 #else
870             signal(SIGALRM, (void*)(int))feeding);
871 #endif /* _NO_PROTO */
872             return;
873         }
874         begone();
875         printf("Unable to read statistics: Error code %d\n
[ %s]\n",
876             SpmiErrno, SpmiErrmsg);
877         must_exit();
878     }
879     loopcnt--;
880 #ifdef MAMFELT
881     if (a2ptx_out)
882     {
883         struct tm *gmt;
884         gmt = gmtime(&tv.tv_sec);
885         fprintf(a2ptx_out, "%04d/%02d/%02d %02d:%02d:%02d",
886             gmt->tm_year + 1900,
887             gmt->tm_mon + 1,
888             gmt->tm_mday,
889             gmt->tm_hour,
890             gmt->tm_min,
891             gmt->tm_sec);
892     }
893 #endif
894     for (i = 0; i < tabsize; i++)
895     {

```

Figure 136. Code to Print the Timestamp. Placement is Critical

```

#ifdef MAMFELT
FILE   *a2ptx_out = NULL;
FILE   *debug = NULL;
boolean silent = FALSE;

datavalue(float v)
{
    if (a2ptx_out)
    {
        if (v < 0.0)
            fprintf(a2ptx_out, " -");
        else
            fprintf(a2ptx_out, " %1.2f", v);
    }
}
#endif

843 void feeding()
844 {
880     for (i = 0; i < tabsize; i++)
881     {
#ifdef MAMFELT
            v = SpmiGetValue(ssp, svp[i]) * prt1[i].mul;
#else
            f = SpmiGetValue(ssp, svp[i]);
            v = f * prt1[i].mul;
            datavalue(f);
#endif
882     v = SpmiGetValue(ssp, svp[i]) * prt1[i].mul;
883     if (v >= 0)
884     {
885         prt1[i].fun(v, prt1[i].line, prt1[i].col);
905     }

1000 }

```

Figure 137. Code Added to Print the Actual Data Values

An alternate approach to just printing floats for every value would be to check the ValType of the data first. See Figure 138 for a sample of how you can do this. We used this routine to debug the Spmi values we were receiving.

```

my_debug(int i)
{
    /*
     * parameter i is index into statset array svp
     */
    struct SpmiStat      *spmistat;
    float   value;

    if (a2ptx_out)
    {
        value = SpmiGetValue(ssp, svp[i]);
        spmistat = SpmiGetStat((svp[i])->stat);
        fprintf(debug, "%2d. %c%02d%-10s %-20s: %g\n",
                i+1,
                spmistat->value_type == SiCounter ? 'C' : 'L',
                spmistat->data_type,
                spmistat->name,
                spmistat->description,
                value);
    }
}

```

Figure 138. A Code Sample for Output Dependent on ValType of Statistic

Appendix C. Performance Toolbox (PTX) Fileset Contents

As pointed out, PTX and PTPE consist of several filesets. The contents of these filesets are listed here.

C.1 Manager

The manager code consists of the following components:

| | |
|-----------------|--|
| xmperf | The main user interface program providing graphical display of local and remote performance information and a menu interface to commands of your choice. |
| 3dmon | A program that can monitor up to 576 statistics simultaneously and display the statistics in a three-dimensional graph. |
| 3dplay | A program to playback 3dmon recording files in a 3dmon-like view. This program is available in Version 2.2 or later. |
| exmon | A graphical program to monitor exception conditions generated by filtd daemon running on local or remote. |
| azizo | A program that allows you to analyze any recording of performance data. It lets you zoom-in on sections of the recording and provides graphical as well as tabular views of the entire recording or zoomed-in parts of it. |
| chmon | Supplied as an executable as well as in source form, this program allows monitoring of vital statistics from a character terminal. |
| a2ptx | The a2ptx program can generate recordings from ASCII files in a format as produced by the ptxtab or ptxrlog programs or the Performance Toolbox for AIX SpmiLogger sample program. The generated recording can then be played back by xmperf or analyzed with azizo. |
| ptxconv | The format of recordings has changed between versions of the Performance Toolbox for AIX. As a convenience to users of multiple versions of the Performance Toolbox for AIX, this program converts recording files between the formats of the different versions. |
| ptxls | A program to list the control information of a recording file, including a list of the statistics defined in the file. |
| ptxmerge | This program allows you to merge up to 10 recording files into one. For example, you could merge xmserverd recordings from the client and server sides of an application into one file to better correlate the performance impact of the application on the two sides. |
| ptxrlog | A program to create recordings in ASCII or binary format. |
| ptxsplit | In cases where recording files are too large to analyze as one file, this program allows you to split the file into multiple smaller files for better overview and faster analysis. |
| ptxtab | A program to tabulate the contents of a recording file for printed output. This tool was previously called xmtab. |

| | |
|--|--|
| ptx2stat | Converts HotSet data collected in a recording file to a format that resembles the recording format for StatSets. Permits postprocessing of HotSet data with the programs that allow playback and manipulation of recordings. |
| ptxhottab | A program that can format and print HotSet information collected in recording files. |
| Samples | Sample data consumer programs that illustrate the use of the API. The chmon command is simply a curses-based activity monitor. |
| Remote Statistics Interface API | Header file to allow you to develop your own data consumer programs. |

C.2 Agent

The agent code consists of the following components:

| | |
|---|---|
| xmservd | The data supplier daemon, which permits a system where this daemon is running to supply performance statistics to data-consumer programs on the local or remote hosts. This daemon also provides the interface to SNMP. Note: The interface to SNMP is available only on RS/6000 Agents. |
| xmscheck | A program that lets you pre-check the xmservd recording configuration file. This program is very useful when you want to start and stop xmservd recording at predetermined times. |
| xmpeek | A program that allows you to display the status of xmservd on the local or remote hosts and to list all available statistics from the daemon. |
| filtd | A daemon that can be used to do data reduction of existing statistics and to define alarm conditions and the alarm triggers. |
| iphosts | A program to initiate monitoring of Internet Protocol performance by specifying which hosts to monitor. Accepts a list of hosts from the command line or from a file. |
| armtoleg | A program that can convert a preexisting Application Response Measurement (ARM) library into an ARM library that can be accessed concurrently with the ARM library shipped with PTX. Only required and available on AIX systems. |
| SpmiArmd | A daemon that collects Application Response Measurement (ARM) data and interfaces to the Spmi library code to allow monitoring of ARM metrics from any PTX Manager program. |
| SpmiResp | A daemon that polls for IP response times for selected hosts and interfaces to the Spmi library code to allow monitoring of IP response time metrics from any PTX Manager program. |
| System Performance Measurement Interface API and Library | Header files and a library to allow you to develop your own dynamic data supplier and local data consumer programs. |

Application Response Measurement API and Libraries

A header file and two libraries support the PTX implementation of ARM. The implementation allows for coexistence and

simultaneous use of the PTX ARM library and one previously installed ARM library.

Samples Sample dynamic data supplier and data consumer programs that illustrate the use of the API. These samples are provided in source code only and are located in `/usr/samples/perfagent/server`. A program `lchmon` provides the same functionality as `chmon`, but only for the local system.

PTX Agents for Selected Non-IBM Platforms

Compressed tar files provide full agent support on multiple HP-UX and Solaris versions on a variety of HP and Sun systems.

C.3 PTPE Filesets

The PTPE components are grouped into three major filesets. This allows for greater control when selecting which elements to install for each type of SP node in your environment. By planning carefully, you can minimize the amount of disk space needed to support PTPE operations.

C.3.1 The `ptpe.program` Image

This contains all the monitoring daemons and configuration files and **MUST** be installed on any SP node that you plan to monitor.

| | |
|-----------------|---|
| spdmmd | PTPE master daemon scheduled by <code>inetd</code> |
| spdmcold | Data collection and summarization daemon |
| spdmspld | Performance statistic collection and archiving daemon |
| spdmapid | API manager daemon for data and control functions |
| spdmtrmd | Termination daemon for shutdown and cleanup |

C.3.2 The `ptpe.gui` Fileset

This fileset contains the code and configuration files for the SP Performance Monitor graphical user interface extension to *Perspectives*, and should be installed on those nodes where you intend to use the Motif interface functions.

| | |
|------------------|-------------------------------|
| spperfmon | PTPE performance monitor GUI. |
|------------------|-------------------------------|

C.3.3 The `ptpe.docs` Fileset

This fileset contains the PTPE documentation files in PostScript, man, and info formats for online and offline reference. This fileset could be installed on a single node and shared through the Network File System (NFS) or replicated through `supper` to other nodes that need it. The combined files occupy about 9 MB in the `man`, `info2` and `docs` subdirectories.

C.4 PTPE Files

The directories, files and processes used by PTPE are described here. The following files, commands, and directories are created when PTPE is installed.

C.4.1 Files, Directories and Libraries

The following files, directories and libraries are installed:

/usr/lpp/ptpe

Location of the PTPE installable image, commands, and daemons.

/var/adm/ptpe

Location of the PTPE information archive for the node.

/var/adm/ptpe/perftab

The PTPE archived statistics translation table. This file contains the mapping between the Spmi names for performance statistics and the identification codes used for these statistics in the performance information archive on the node.

/usr/include/spdm.h

The PTPE C language header file, providing definitions of the data types and prototypes of the PTPE API library subroutines.

/usr/lib/libptpe.a

The PTPE C language programming interface.

/usr/lpp/ptpe/samples

Sample API and configuration files.

/usr/lpp/ssp/info2

PTPE hypertext information database files for InfoExplorer retrieval.

/usr/lpp/ssp/docs/sp_perf_parallel_ext.ps

This manual in printable PostScript format.

Manual pages for PTPE commands and subroutines are included with Parallel System Support Programs man pages in `/usr/lpp/ssp/man`.

C.4.2 Commands and Utilities

The following commands and utilities are installed:

/usr/sbin/ptpectrl

The control command for PTPE. This command controls the current status of performance information sampling, collection and recording in the entire system.

/usr/lpp/ptpe/bin/ptpeconf

The configuration command for PTPE. This command creates the necessary data classes within the System Data Repository.

/usr/lpp/ptpe/bin/ptpegroup

A user group creation command, which creates the perfmom user group. Only users of this group, with the group set as their primary group, can execute any PTPE commands or use PTPE's programming library.

/usr/lpp/ptpe/bin/ptpehier

The hierarchy construction command of PTPE. This command permits the user to create a monitoring hierarchy by using a set of standard methods or using input provided by the caller.

/usr/lpp/ptpe/bin/ptpedump

A distributed performance information archive dump utility. This utility dumps a text version of the archives maintained by one or more nodes to files on these systems.

/usr/lpp/ptpe/bin/spdm_dump

A local performance information archive dump utility. This utility dumps a text version of the archive maintained by the local node to standard output.

C.4.3 Daemons

The following daemons are installed:

/usr/lpp/ptpe/bin/spdmd

The PTPE master daemon. This daemon receives all PTPE requests for the node, validates the request, and invokes the appropriate daemon process to handle the request.

/usr/lpp/ptpe/bin/spdmcold

The PTPE collector daemon. This daemon is executed on the central coordinator node and on all data manager nodes in the monitoring hierarchy and prepares the averaged performance statistics for the monitoring hierarchy.

/usr/lpp/ptpe/bin/spdmspld

The PTPE sampler daemon. This daemon is executed on all nodes within the monitoring hierarchy. This daemon obtains all the performance information from the node, forwards this information to the node's data manager node, and records the information to that node's performance information archive.

/usr/lpp/ptpe/bin/spdmapid

The PTPE programming library request handler. This daemon executes whenever the node is involved in the response to a programming library request. It obtains performance data recorded in the archives of a node, enables or restricts information from being collected or archived, and (on data manager nodes) relays requests on to other nodes for further processing.

/usr/lpp/ptpe/bin/spdmtrmd

The PTPE termination daemon. This daemon executes on a node whenever performance information collection is being shut down and is responsible for forcing the other PTPE daemons to exit.

C.4.4 Message Catalogs

The following message catalogs are installed:

/usr/lib/nls/msg/En_US/ptpe.cat

/usr/lib/nls/msg/en_US/ptpe.cat

/usr/lib/nls/msg/C/ptpe.cat

The message catalog for Performance Toolbox Parallel Extensions for AIX. Other message catalogs may be provided in other directories for other locales.

C.4.5 What PTPE Creates During Use

These files are not created until PTPE is started.

/var/adm/ptpe/perflog

The PTPE information archive. This file is created on each node in the monitoring hierarchy after collection has begun. The file contains a statistics code translation table plus any performance information that was recorded by this node.

/etc/perf/spdm.pid

This file contains the process identifiers of any PTPE daemons currently executing on the node. This file should not remain after performance information collection has been shut down.

/etc/perf/ptpe.shseg

A file used by the Spmi library when creating the shared memory to store the SP-specific performance data. This file should not remain after performance information collection has been shut down or until the `ptperrm` command has completed.

/tmp/spdm.trace

The PTPE daemon tracing file. This file is only present if the daemons encounter unexpected error conditions. The messages in this file should also be mirrored in the node's error log.

/tmp/manager.cont.file

/tmp/manager.cont.file.uniq

/tmp/manager.cont.file.sorted

/tmp/manager.stat.file

/tmp/manager.stat.file.uniq

/tmp/manager.stat.file.sorted

/tmp/manager.stat.file.transposed

/tmp/current.stats

Work files used by the PTPE daemons when initializing the monitoring hierarchy. The files should only exist on the node during the initialization phase and should be removed after that phase has completed.

Appendix D. Special Notices

This publication is intended to help anyone who needs to customize the monitoring and analysis of the Performance Toolbox for AIX (PTX) and the Performance Toolbox Parallel Extension (PTPE). The information in this publication is not intended as the specification of any programming interfaces that are provided by the Performance Toolbox for AIX and the Performance Aide for AIX Licensed Programs or by the Performance Parallel Extensions for AIX, a feature of IBM Parallel System Support Programs for AIX. See the PUBLICATIONS section of the IBM Programming Announcement for these products 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 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.

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:

AIX®
IBM®
LoadLeveler®
POWERparallel®
RS/6000

DB2®
InfoExplorer
NetView®
RISC System/6000®
SP

The following terms are trademarks of other companies:

Microsoft, Windows, Windows NT, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

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

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

Appendix E. Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

E.1 International Technical Support Organization Publications

For information on ordering these ITSO publications see "How to Get ITSO Redbooks" on page 203

- *RS/6000 Performance Tools in Focus*, SG24-4989
- *Understanding IBM RS/6000 Performance and Sizing*, SG24-4810
- *RS/6000 SP High Availability Infrastructure*, SG24-4838

E.2 Redbooks on CD-ROMs

Redbooks are also available on CD-ROMs. **Order a subscription** and receive updates 2-4 times a year at significant savings.

| CD-ROM Title | Subscription Number | Collection Kit Number |
|---|---------------------|-----------------------|
| System/390 Redbooks Collection | SBOF-7201 | SK2T-2177 |
| Networking and Systems Management Redbooks Collection | SBOF-7370 | SK2T-6022 |
| Transaction Processing and Data Management Redbook | SBOF-7240 | SK2T-8038 |
| AS/400 Redbooks Collection | SBOF-7270 | SK2T-2849 |
| RS/6000 Redbooks Collection (HTML, BkMgr) | SBOF-7230 | SK2T-8040 |
| RS/6000 Redbooks Collection (PostScript) | SBOF-7205 | SK2T-8041 |
| Application Development Redbooks Collection | SBOF-7290 | SK2T-8037 |
| Personal Systems Redbooks Collection | SBOF-7250 | SK2T-8042 |

E.3 Other Publications

These publications are also relevant as further information sources:

- *Performance Tuning Guide*, SC23-2365
- *Performance Toolbox for AIX: Guide and Reference*, SC23-2625
- *Performance Toolbox Parallel Extensions for AIX: Guide and Reference*, SC23-3997
- *IBM PSSP 2.2 Administration Guide*, GC23-3897
- *IBM PSSP 2.2 Installation and Migration*, GC23-3898
- *IBM PSSP 2.2 Command and Technical Reference*, GC23-3900

How to Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, CD-ROMs, workshops, and residencies. A form for ordering books and CD-ROMs is also provided.

This information was current at the time of publication, but is continually subject to change. The latest information may be found at <http://www.redbooks.ibm.com>.

How IBM Employees Can Get ITSO Redbooks

Employees may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **PUBORDER** — to order hardcopies in United States
- **GOPHER link to the Internet** - type GOPHER.WTSCPOK.ITSO.IBM.COM
- **Tools disks**

To get LIST3820s of redbooks, type one of the following commands:

```
TOOLS SENDTO EHONE4 TOOLS2 REDPRINT GET SG24xxxx PACKAGE
TOOLS SENDTO CANVM2 TOOLS REDPRINT GET SG24xxxx PACKAGE (Canadian users only)
```

To get BookManager BOOKs of redbooks, type the following command:

```
TOOLCAT REDBOOKS
```

To get lists of redbooks, type one of the following commands:

```
TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET ITSOCAT TXT
TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET LISTSERV PACKAGE
```

To register for information on workshops, residencies, and redbooks, type the following command:

```
TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ITSOREGI 1996
```

For a list of product area specialists in the ITSO: type the following command:

```
TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ORGCARD PACKAGE
```

- **Redbooks Web Site on the World Wide Web**

<http://w3.itso.ibm.com/redbooks>

- **IBM Direct Publications Catalog on the World Wide Web**

<http://www.elink.ibm.com/pb1/pb1>

IBM employees may obtain LIST3820s of redbooks from this page.

- **REDBOOKS category on INEWS**
- **Online** — send orders to: USIB6FPL at IBMMAIL or DKIBMBSH at IBMMAIL
- **Internet Listserver**

With an Internet e-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an e-mail note to announce@webster.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank). A category form and detailed instructions will be sent to you.

Redpieces

For information so current it is still in the process of being written, look at "Redpieces" on the Redbooks Web Site (<http://www.redbooks.ibm.com/redpieces.htm>). 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.

How Customers Can Get ITSO Redbooks

Customers may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Online Orders** — send orders to:

| | | |
|------------------------|---------------------------------------|---|
| In United States: | IBMMAIL usib6fpl at ibmmail | Internet usib6fpl@ibmmail.com |
| In Canada: | caibmbkz at ibmmail | lmannix@vnet.ibm.com |
| Outside North America: | dkibmbsh at ibmmail | bookshop@dk.ibm.com |

- **Telephone orders**

| | |
|---------------------------|-------------------------------|
| United States (toll free) | 1-800-879-2755 |
| Canada (toll free) | 1-800-IBM-4YOU |
| Outside North America | (long distance charges apply) |
| (+45) 4810-1320 - Danish | (+45) 4810-1020 - German |
| (+45) 4810-1420 - Dutch | (+45) 4810-1620 - Italian |
| (+45) 4810-1540 - English | (+45) 4810-1270 - Norwegian |
| (+45) 4810-1670 - Finnish | (+45) 4810-1120 - Spanish |
| (+45) 4810-1220 - French | (+45) 4810-1170 - Swedish |

- **Mail Orders** — send orders to:

| | | |
|--|--|--|
| IBM Publications Publications Customer Support P.O. Box 29570 Raleigh, NC 27626-0570 USA | IBM Publications 144-4th Avenue, S.W. Calgary, Alberta T2P 3N5 Canada | IBM Direct Services Sortemosevej 21 DK-3450 Allerød Denmark |
|--|--|--|

- **Fax** — send orders to:

| | |
|---------------------------|---|
| United States (toll free) | 1-800-445-9269 |
| Canada | 1-403-267-4455 |
| Outside North America | (+45) 48 14 2207 (long distance charge) |

- **1-800-IBM-4FAX (United States) or (+1)001-408-256-5422 (Outside USA)** — ask for:

Index # 4421 Abstracts of new redbooks
Index # 4422 IBM redbooks
Index # 4420 Redbooks for last six months

- **Direct Services** - send note to softwareshop@vnet.ibm.com

- **On the World Wide Web**

| | |
|---------------------------------|---|
| Redbooks Web Site | http://www.redbooks.ibm.com |
| IBM Direct Publications Catalog | http://www.elink.ibm.com/pbl/pbl |

- **Internet Listserver**

With an Internet e-mail address, anyone can subscribe to an IBM Announcement Listserv. To initiate the service, send an e-mail note to announce@webster.ibm.com with the keyword `subscribe` in the body of the note (leave the subject line blank).

Redpieces

For information so current it is still in the process of being written, look at "Redpieces" on the Redbooks Web Site (<http://www.redbooks.ibm.com/redpieces.htm>). 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.

IBM Redbook 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

| | | | |
|---------------|---|----------------|--|
| 3-D | Three Dimensional | NetIF | Network Interface |
| AIX | Advanced Interactive Executive | NFS | Network File System |
| API | Application Program Interface | NIM | Network Installation Manager |
| ARM | Application Response Measurement | PAIDE | Performance Aide |
| ASCII | American National Standard Code for Information Interchange | PagSp | Paging Space |
| DB | Data Base | PE | Parallel Extension |
| CD-ROM | Compact Disk Read Only Memory | PSSP | Parallel System Support Programs |
| CPU | Central Processing Unit | PTF | Program Temporary Fix |
| CWS | Control Workstation | PTPE | Performance Toolbox Parallel Extensions |
| DDS | Dynamic Data Supplier | PTX | Performance Toolbox |
| FS | Filesystem | RISC | Reduced Instruction Set Computer |
| GID | Group Identification | RS/6000 | RISC System/6000 |
| GUI | Graphical User Interface | Rsi | Remote Statistics Interface |
| IBM | International Business Machines Corporation | SAS | Structural Analysis System |
| ICMP | Internet Control Management Protocol | SDR | System Data Repository |
| ID | Identification | SNMP | Simple Network Management Protocol |
| IP | Internet Protocol | SMUX | Single Multiplexor |
| ISS | Information Service System | SP | Scalable POWERparallel |
| ITSO | International Technical Support Organization | Spmi | System Performance Measurement Interface |
| JFS | Journaled File System | SUID | Set-user-ID-on-execution |
| kmem | Kernel Memory | SysIO | System Input/Output |
| LAN | Local Area Network | UDB | Universal Data Base |
| LPP | Licensed Program Product | UDP | User Datagram Protocol |
| MIB | Management Information Base | UID | User Identification |
| | | VSD | Virtual Shared Disk |
| | | X | X Window System |

Index

Numerics

3dmon 51, 53, 96, 152, 161
Options 53
3dmon.cf 54, 61
3dplay 65, 103
3dplay.hlp 66

A

abbreviations 207
acronyms 207
Agent 3
Alarms 83, 88
analy 38
Analysis 166
Annotation 95, 97, 105, 112
Application Response Measurement (ARM) 90, 164
Archiving Data 147
ARM 90, 164
azizo 105

B

bibliography 201

C

Configuration
3Dmon 53
3dmon.cf 54, 61
3Dplay 65
3dplay.hlp 66
Advanced 51
Basic 7
EXmon 66
exmon.cf 69
exmon.hlp 70
filter.cf 81
Menu Bar 33
Process Controls 43
Resptime.cf 89
Rsi.host 13
SpmiArmd.cf 90
XMperf 7
xmperf.cf 14, 19
xmperf.hlp 45
xmservd.cf 76, 99
xmservd.res 74
Console 17
Default 17
Keywords 19
ctrl 41

D

Data Collection 146
DDS 75
Dynamic Data Suppliers (DDS) 75

E

Exceptions 83, 86
exmon 66, 86, 159
exmon.cf 69
exmon.hlp 70

F

Filesets 3, 121, 193
filtld 81
Options 81
Stopping 81
filter.cf 81, 83
Filtering 110

G

Graphs
Recording 30
State 31

H

Help 45
3dplay.hlp 66
exmon.hlp 70
xmperf.hlp 46
Hierarchy 116, 125, 140
Guidelines 125
Specifications 132
HotSets 78, 169, 170

I

Instruments
Local 19
Remote 23
Skeleton 25
IP Response Time 89, 163
iphosts 89

L

lchmon 183
Limit Access 74
lmon 183, 188

M

Manager 3
 Menu Bar 33
 Analysis 38
 Controls 41
 Monitor 24
 Utilities 34
 Monitoring 159
 3dmon 161
 ARM 164
 exmon 159
 IP Response Time 163
 Lotus Notes Servers 164
 xmperf 163

P

pa6000 88
 Performance Analyzer/6000 88
 Perspectives 134, 141
 Playback 99
 3dplay 65, 103
 azizo 105
 xmperf 99
 Postprocessing 166
 proc 44
 Process Controls 43
 PTPE 3, 115
 3dmon 152
 Archive Space 129
 Archiving Data 147
 Configuration Files 134
 Coordinator 115, 120
 Daemons 197
 Data 117
 Data Collection 146
 Filesets 122, 195
 Hierarchy 116, 125, 132, 140
 Installing 126
 Interfaces 134
 Logs 129
 Manager 115, 120
 Perspectives 141
 Preparation 119
 Reporter 115, 120
 User 128
 xmperf 150
 ptpe.cf 134
 ptpelier 132, 134
 ptx2stat 170
 ptxhottab 169
 ptxls 167
 ptxrlog 97
 ptxsplitt 168
 ptxtab 167, 172

R

Recording 93
 3dmon 96
 ptxrlog 97
 xmperf 93
 xmservd 99
 Resptime.cf 89
 Rsi.hosts 13
 Runtime Control 91

S

Search Path 7, 14, 46, 51, 74, 81, 117
 SMPI 89
 SNMP 75, 87
 SpmiArmd 90
 SpmiArmd.cf 90
 SpmiResp 89
 Spreadsheets 172

T

Threshold 88
 Tools 3
 TRAP 87

U

util 34

W

Wildcard 25, 54, 57, 59, 62, 63
 Wildcards 82

X

xmpeek 61, 91, 135
 XMperf 7, 15, 93, 99, 150, 163
 Colors 9
 Consoles 17
 Customization 11
 File 7
 Fonts 7
 Local Instruments 19
 Options 15
 Patterns 11
 Remote Instruments 23
 Shading 10
 Skeleton 25
 xmperf.cf 14, 19
 xmperf.hlp 45, 46
 xmscheck 79
 xmservd 73, 99, 136
 Options 73
 Refresh 76
 xmservd.cf 76, 99
 xmservd.res 74, 89, 136

ITSO Redbook Evaluation

Customizing Performance Toolbox and Performance Toolbox Parallel Extensions for AIX
SG24-2011-00

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.com>
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@vnet.ibm.com

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!)



This soft copy for use by IBM employees only.

Printed in U.S.A.

SG24-2011-00

