

# **PATROL<sup>®</sup> Agent Reference Manual**

**Version 3.5**

**January 31, 2003**



Copyright 2003 BMC Software, Inc., as an unpublished work. All rights reserved.

BMC Software, the BMC Software logos, and all other BMC Software product or service names are registered trademarks or trademarks of BMC Software, Inc. All other registered trademarks or trademarks belong to their respective companies.

THE USE AND CONTENTS OF THIS DOCUMENTATION ARE GOVERNED BY THE SOFTWARE LICENSE AGREEMENT ENCLOSED AT THE BACK OF THIS DOCUMENTATION.

## Restricted Rights Legend

U.S. GOVERNMENT RESTRICTED RIGHTS. UNPUBLISHED—RIGHTS RESERVED UNDER THE COPYRIGHT LAWS OF THE UNITED STATES. Use, duplication, or disclosure by the U.S. Government is subject to restrictions set forth in FAR Section 52.227-14 Alt. III (g)(3), FAR Section 52.227-19, DFARS 252.227-7014 (b), or DFARS 227.7202, as amended from time to time. Contractor/Manufacturer is BMC Software, Inc., 2101 CityWest Blvd., Houston, TX 77042-2827, USA. Any contract notices should be sent to this address.

---

## Contacting BMC Software

You can access the BMC Software Web site at <http://www.bmc.com>. From this Web site, you can obtain information about the company, its products, corporate offices, special events, and career opportunities.

### United States and Canada

**Address** BMC Software, Inc.  
2101 CityWest Blvd.  
Houston TX 77042-2827

**Telephone** 713 918 8800 or  
800 841 2031

**Fax** 713 918 8000

### Outside United States and Canada

**Telephone** (01) 713 918 8800

**Fax** (01) 713 918 8000

---

## Customer Support

You can obtain technical support by using the Support page on the BMC Software Web site or by contacting Customer Support by telephone or e-mail. To expedite your inquiry, please see “Before Contacting BMC Software.”

### Support Web Site

You can obtain technical support from BMC Software 24 hours a day, 7 days a week at <http://www.bmc.com/support.html>. From this Web site, you can

- read overviews about support services and programs that BMC Software offers
- find the most current information about BMC Software products
- search a database for problems similar to yours and possible solutions
- order or download product documentation
- report a problem or ask a question
- subscribe to receive e-mail notices when new product versions are released
- find worldwide BMC Software support center locations and contact information, including e-mail addresses, fax numbers, and telephone numbers

### Support by Telephone or E-mail

In the United States and Canada, if you need technical support and do not have access to the Web, call 800 537 1813. Outside the United States and Canada, please contact your local support center for assistance. To find telephone and e-mail contact information for the BMC Software support center that services your location, refer to the Contact Customer Support section of the Support page on the BMC Software Web site at [www.bmc.com/support.html](http://www.bmc.com/support.html).

### Before Contacting BMC Software

Before you contact BMC Software, have the following information available so that Customer Support can begin working on your problem immediately:

- product information
  - product name
  - product version (release number)
  - license number and password (trial or permanent)
- operating system and environment information
  - machine type
  - operating system type, version, and service pack or other maintenance level such as PUT or PTF
  - system hardware configuration
  - serial numbers
  - related software (database, application, and communication) including type, version, and service pack or maintenance level

- sequence of events leading to the problem
- commands and options that you used
- messages received (and the time and date that you received them)
  - product error messages
  - messages from the operating system, such as `file system full`
  - messages from related software

---

## Contents

|                  |  |      |
|------------------|--|------|
| <b>Chapter 1</b> | <b>Structure of PATROL</b>                                     |      |
|                  | Role of the PATROL Agent . . . . .                             | 1-2  |
|                  | Purpose of PATROL . . . . .                                    | 1-2  |
|                  | Overview of the PATROL 3.x Architecture . . . . .              | 1-3  |
|                  | Overview of the PATROL 7.x Architecture . . . . .              | 1-4  |
|                  | Components of PATROL . . . . .                                 | 1-6  |
|                  | PATROL Components . . . . .                                    | 1-6  |
|                  | Non-PATROL Components . . . . .                                | 1-9  |
|                  | PATROL Agent Directory and File Structure . . . . .            | 1-10 |
|                  | <i>PATROL_HOME</i> Directory Structure . . . . .               | 1-10 |
|                  | Files Used by the PATROL Agent . . . . .                       | 1-11 |
|                  | Event Management with the PATROL Agent . . . . .               | 1-14 |
|                  | Services Provided by the PATROL Agent . . . . .                | 1-14 |
|                  | Overview of the PEM Engine . . . . .                           | 1-16 |
|                  | Management Services from the PEM Engine . . . . .              | 1-17 |
| <br>             |  |      |
| <b>Chapter 2</b> | <b>Starting and Stopping the PATROL Agent</b>                  |      |
|                  | Methods for Starting the PATROL Agent . . . . .                | 2-3  |
|                  | Default Port Number . . . . .                                  | 2-3  |
|                  | Startup Process . . . . .                                      | 2-3  |
|                  | Command Line Arguments for Starting the PATROL Agent . . . . . | 2-4  |
|                  | Methods for Stopping the PATROL Agent . . . . .                | 2-8  |
|                  | Stopping the PATROL Process . . . . .                          | 2-8  |
|                  | Improper Shutdown . . . . .                                    | 2-9  |
|                  | PATROL Configuration Utility (pconfig) . . . . .               | 2-10 |
|                  | Starting the PATROL Agent on Unix . . . . .                    | 2-11 |
|                  | Starting the PATROL Agent for Unix . . . . .                   | 2-12 |
|                  | Stopping the PATROL Agent on Unix . . . . .                    | 2-13 |

|   |      |
|---|------|
| Stopping the PATROL Agent for Unix with a Script . . . . .                          | 2-14 |
| Stopping the PATROL Agent for Unix with a<br>Configuration Utility Script . . . . . | 2-15 |
| Starting the PATROL Agent on Windows . . . . .                                      | 2-16 |
| PATROL Agent Configuration Utility . . . . .  | 2-16 |
| Modes for Running the PATROL Agent for Windows . . . . .                            | 2-17 |
| Specifying an Account for the PATROL Agent for Windows . . . . .                    | 2-18 |
| Starting the PATROL Agent for Windows with<br>Services Applet . . . . .             | 2-20 |
| Starting the PATROL Agent for Windows from<br>Command Prompt . . . . .              | 2-21 |
| Active Directory Support . . . . .  | 2-22 |
| Publishing to Active Directory . . . . .  | 2-22 |
| Information Stored in Active Directory . . . . .                                    | 2-23 |
| Stopping the PATROL Agent on Windows . . . . .                                      | 2-24 |
| Stopping the PATROL Agent for Windows with<br>Services Applet . . . . .             | 2-25 |
| Stopping the PATROL Agent for Windows from<br>the Command Line . . . . .            | 2-26 |
| Starting the PATROL Agent on OpenVMS . . . . .                                      | 2-27 |
| Syntax of PATROL\$STARTUP.COM . . . . .   | 2-27 |
| Batch Process Startup of the PATROL Agent on OpenVMS . . . . .                      | 2-27 |
| Starting the PATROL Agent on OpenVMS from<br>the Command Line . . . . .             | 2-28 |
| Starting the PATROL Agent on OpenVMS in a Batch Process . . . . .                   | 2-29 |
| Stopping the PATROL Agent on OpenVMS . . . . .                                      | 2-30 |
| Syntax for PATROL\$SHUTDOWN.COM . . . . .   | 2-30 |
| Stop the PATROL Agent on OpenVMS in a Batch Process . . . . .                       | 2-30 |
| Stopping the PATROL Agent on OpenVMS from the<br>Command Prompt . . . . .           | 2-31 |
| Stopping the PATROL Agent on OpenVMS in a Batch Process . . . . .                   | 2-32 |

### Chapter 3

#### Background About Configuring the PATROL Agent

|  |     |
|--|-----|
| Controlling the Configuration of PATROL Agents . . . . .         | 3-2 |
| Essential Controls for Configuring PATROL Agents . . . . .       | 3-2 |
| Overriding the Default Values . . . . .                          | 3-5 |
| Override Parameters and the PATROL Agent Configuration . . . . . | 3-6 |
| Variables that Cannot Be Changed . . . . .                       | 3-7 |
| When Changes to the Configuration Take Effect . . . . .          | 3-7 |
| Process for Configuring the PATROL Agent . . . . .               | 3-9 |

|   |      |
|---|------|
| Methods for Specifying Files and Changing Variables . . . . . | 3-10 |
| Options for Changing Values for Variables . . . . .           | 3-11 |
| Determining the Method for Changing Variables . . . . .       | 3-11 |
| Creating User-Defined Variables . . . . .                     | 3-12 |
| Displaying License File Expiration Warnings . . . . .         | 3-12 |

**Chapter 4      Establishing Accounts and Ports**

|   |     |
|---|-----|
| Accounts . . . . .  | 4-2 |
| Setting the PATROL Agent Default Account . . . . .            | 4-2 |
| Setting the PATROL Agent Default Account Shell . . . . .      | 4-2 |
| Setting the PATROL Agent Account for Applications . . . . .   | 4-3 |
| Adding Time Zones to PATROL . . . . .                         | 4-4 |
| Setting the PATROL Agent Account for Instances . . . . .      | 4-4 |
| Default Accounts for XPC Servers . . . . .                    | 4-5 |
| Using the Application-Specific Account for Commands . . . . . | 4-5 |
| Setting the Default Account for Trusted Clients . . . . .     | 4-7 |
| Ports . . . . .   | 4-8 |
| Setting the Default Port Number on Unix Only . . . . .        | 4-8 |
| Setting a Local Port for Remote Calls . . . . .               | 4-9 |

**Chapter 5      Managing Console Connections**

|   |      |
|---|------|
| The Agent-Console Relationship . . . . .  | 5-2  |
| Displaying and Logging PEM Connection Messages . . . . .                                      | 5-2  |
| Changing the Unix Display Environment . . . . .   | 5-3  |
| Controlling System Output Window Display . . . . .  | 5-3  |
| Setting the Agent to Run without a Console Connection . . . . .                               | 5-4  |
| Designating an IP Address for Connections on Systems with<br>Multiple Network Cards . . . . . | 5-5  |
| Recognize Additional IP Addresses . . . . .   | 5-7  |
| Connections Using DHCP . . . . .  | 5-8  |
| Controlling Access to the Agent . . . . .   | 5-9  |
| Backing Up Your Configuration Files Before Defining an ACL . . . . .                          | 5-9  |
| Defining Access Control Lists . . . . .   | 5-9  |
| Enabling System Output Window Display . . . . .   | 5-13 |
| HostName and UserName Attribute Conventions . . . . .   | 5-13 |
| Connection Modes and Accounts . . . . .   | 5-15 |
| Required Access . . . . .   | 5-16 |
| Restoring Access in Case of Lockout . . . . .   | 5-16 |
| Examples of ACL Usage . . . . .   | 5-17 |

|   |      |
|---|------|
| Access for Agent Configuration Utility Started through the<br>Developer Console ..... | 5-18 |
| Access for Agent Configuration Utility Started from the<br>Command Line .....         | 5-19 |
| Access for PATROL Event Manager Console .....   | 5-19 |
| Access for the Knowledge Module Scripts .....   | 5-20 |
| Access for PEM Applications .....   | 5-21 |
| Error Messages .....  | 5-21 |
| Controlling Events Displayed in PEM .....   | 5-21 |

## Chapter 6

### Support of Clusters and Failovers

|  |      |
|--|------|
| Cluster and Failover Concepts .....                            | 6-3  |
| Control Script .....   | 6-3  |
| Cluster .....  | 6-3  |
| Cluster Node .....   | 6-3  |
| Cluster Application .....                                      | 6-3  |
| Cluster Management Software (CMS) .....                        | 6-4  |
| Failover .....   | 6-4  |
| Group .....  | 6-4  |
| Package .....  | 6-5  |
| Virtual IP Address .....                                       | 6-5  |
| Examples of Third Party Cluster Software .....                 | 6-5  |
| PATROL Cluster Compatibility and Failover Tolerance .....      | 6-6  |
| Failover Behavior .....  | 6-7  |
| Description .....  | 6-7  |
| Illustration .....   | 6-9  |
| PATROL Agent Issues and Concerns .....                         | 6-12 |
| One Virtual IP Address For Each Group or Package .....         | 6-12 |
| PATROL Must Recognize the Virtual IP Address .....             | 6-12 |
| Local KM Customizations .....                                  | 6-13 |
| Basic Set Up of the PATROL Agent in a Windows Cluster .....    | 6-14 |
| Using the PATROL Cluster Configuration Wizard .....            | 6-14 |
| Manually Configuring the PATROL Agent for Clustering .....     | 6-15 |
| Define the PATROL Cluster-Specific Environment Variables ..... | 6-17 |
| Create and Register a New Service for the PATROL Agent .....   | 6-18 |
| Define the PATROL Agent as a Member of the Group .....         | 6-20 |
| Basic Set Up of the PATROL Agent in a Unix Cluster .....       | 6-23 |
| Configure PATROL Agent to Operate in a Cluster .....           | 6-23 |
| Define the PATROL Cluster-Specific Environment Variables ..... | 6-25 |
| Define the PATROL Agent as a Member of the Package .....       | 6-26 |

|   |      |
|---|------|
| PATROL Cluster-Specific Environment Variables for History and Configuration | 6-28 |
| Variables   | 6-28 |
| Operation   | 6-29 |
| Example   | 6-30 |

## Chapter 7

### Loading and Monitoring Applications

|   |      |
|---|------|
| Loading Knowledge Modules by Console and Agent              | 7-3  |
| When the PATROL Agent Will Not Monitor Applications         | 7-3  |
| Version Arbitration   | 7-4  |
| Application Status  | 7-6  |
| Statuses  | 7-6  |
| Assigning Status  | 7-6  |
| Preloading Applications                                     | 7-7  |
| Based on Application Name                                   | 7-7  |
| Based on Architecture                                       | 7-8  |
| Designating Applications as Static                          | 7-9  |
| Disabling Applications                                      | 7-9  |
| Based on Application Name                                   | 7-10 |
| Based on Architecture                                       | 7-11 |
| Filter Processing Logic For Disabling KMs                   | 7-12 |
| Types   | 7-12 |
| Precedence  | 7-13 |
| Example   | 7-14 |
| Listing Loaded Applications                                 | 7-16 |
| Running dump_km_list  | 7-16 |
| Example   | 7-17 |
| Selecting Which Instances to Monitor                        | 7-18 |
| Choosing an Inclusive or Exclusive Filter                   | 7-18 |
| Editing List to Filter by Application or Regular Expression | 7-19 |

## Chapter 8

### Using pconfig to Configure the PATROL Agent

|  |      |
|--|------|
| Overview of the pconfig Command Line Configuration Utility | 8-2  |
| Prerequisites for Configuring at the Command Line          | 8-2  |
| Syntax of pconfig  | 8-3  |
| Specifying a Windows Domain Account Name                   | 8-5  |
| Determining the Command String to Use for the Task         | 8-5  |
| Determining the Option to Use for the Task                 | 8-9  |
| Examples of Configuring at the Command Line                | 8-10 |

## Chapter 9

### Using xpcnfig (Unix) to Configure the PATROL Agent

|   |      |
|---|------|
| Overview of the xpcnfig Configuration Utility                             | 9-3  |
| Prerequisites for Configuring with the xpcnfig Utility                    | 9-4  |
| Overview of xpcnfig Functions   | 9-5  |
| The xpcnfig Primary Window  | 9-7  |
| Accessing the xpcnfig Primary Window                                      | 9-8  |
| Closing the xpcnfig Primary Window  | 9-10 |
| Cancelling Changes Not Applied  | 9-10 |
| Stopping an Operation   | 9-10 |
| Returning to the xpcnfig Primary Window                                   | 9-10 |
| Selecting a Change File   | 9-12 |
| When Changes Take Effect  | 9-12 |
| Reinitializing the PATROL Agent   | 9-13 |
| Reloading the config.default File and Change File for the<br>PATROL Agent | 9-15 |
| Stopping the PATROL Agent   | 9-17 |
| Handling Change Files   | 9-18 |
| Tasks Available for Handling Change Files                                 | 9-18 |
| Opening an Existing Change File   | 9-19 |
| Creating a New Change File  | 9-21 |
| Sending a Complete Set of Variables                                       | 9-22 |
| Viewing a Change File   | 9-23 |
| Saving a Change File  | 9-24 |
| Applying the Changes to a PATROL Agent                                    | 9-26 |
| Purging the Existing Configuration from a PATROL Agent                    | 9-28 |
| Sending a New License File to a PATROL Agent                              | 9-30 |
| Handling Variables  | 9-32 |
| Handling Variables from the Primary Window                                | 9-32 |
| Adding a Host to the Host List  | 9-33 |
| Adding New Variables to the Change File                                   | 9-36 |
| Deleting a Variable from the Change File                                  | 9-38 |
| Resetting a Variable to Its Default Value                                 | 9-40 |
| Handling Variables from the Edit Variable Dialog Box                      | 9-40 |
| Modifying Variables Using the Edit Variable Dialog Box                    | 9-41 |
| Inserting a New Change Entry in the Edit Variable Dialog Box              | 9-44 |
| Modifying a Change Entry in the Edit Variable Dialog Box                  | 9-46 |
| Deleting a Change Entry from the Edit Variable Dialog Box                 | 9-48 |
| Modifying the Default Account Variable in the Change File                 | 9-49 |

## Chapter 10

### Using wpconfig (Windows) to Configure the PATROL Agent

|  |       |
|--|-------|
| Overview of the wpconfig Configuration Utility             | 10-2  |
| Prerequisites for Using the wpconfig Configuration Utility | 10-3  |
| Overview of wpconfig Functions                             | 10-3  |
| Working with the wpconfig Window                           | 10-6  |
| Accessing the wpconfig Window                              | 10-7  |
| Closing the wpconfig Window                                | 10-8  |
| Creating a New Configuration                               | 10-8  |
| Tasks Available for Using the wpconfig Utility             | 10-8  |
| Retrieving a Configuration from a PATROL Agent             | 10-8  |
| Getting a Configuration from a PATROL Agent                | 10-9  |
| Adding a Variable to a PATROL Agent Configuration          | 10-11 |
| Adding a New Variable                                      | 10-12 |
| Modifying a Variable in a Configuration                    | 10-14 |
| Modifying an Existing Variable                             | 10-15 |
| Modifying the Default Account Variable                     | 10-18 |
| Deleting a Variable from a Configuration                   | 10-20 |
| Deleting an Existing Variable                              | 10-21 |
| Viewing Changes in a Configuration                         | 10-22 |
| Viewing Configuration Changes                              | 10-23 |
| Applying a Configuration to a PATROL Agent                 | 10-25 |
| Applying a Configuration to an Agent                       | 10-26 |

## Chapter 11

### PATROL Agent Logs

|  |       |
|--|-------|
| PATROL Event Log                                 | 11-3  |
| Events Stored                                    | 11-3  |
| Contents   | 11-3  |
| Location   | 11-4  |
| Setting Up the Event Log File and Size           | 11-4  |
| Naming the Event Log File                        | 11-5  |
| Setting the Event Log File Size                  | 11-5  |
| Managing the PATROL Event Cache                  | 11-6  |
| Setting the Cache Size                           | 11-6  |
| Setting the Number of Events Stored in the Cache | 11-7  |
| Extracting Event Data from the PEM Log           | 11-8  |
| Before You Begin                                 | 11-11 |
| Syntax   | 11-11 |
| Options  | 11-12 |
| Examples   | 11-14 |
| Troubleshooting                                  | 11-14 |

|  |       |
|--|-------|
| PATROL Event Archive .....                                 | 11-15 |
| Contents .....   | 11-15 |
| Location .....   | 11-15 |
| Operation .....  | 11-15 |
| Select Events .....  | 11-16 |
| PATROL Agent Error Log .....                               | 11-16 |
| Contents .....   | 11-16 |
| Location .....   | 11-17 |
| Sample .....   | 11-17 |
| Limiting Size By Restricting the Number of Messages .....  | 11-18 |
| Log File Aging .....                                       | 11-18 |
| Log Files for PATROL Single Sign-On for Windows 2000 ..... | 11-20 |
| Agent Audit Log .....                                      | 11-21 |
| Contents .....   | 11-22 |
| Location .....   | 11-22 |
| Setting Up Audit Logging .....                             | 11-23 |
| Keys and Values for the Audit Log Variable .....           | 11-24 |
| Audit Log File Format .....                                | 11-25 |
| Sample Audit Log File .....                                | 11-27 |

## Chapter 12

### **PATROL Agent Parameter History**

|  |       |
|--|-------|
| Overview of PATROL History .....   | 12-3  |
| Contents .....   | 12-3  |
| Structure .....  | 12-4  |
| Location .....   | 12-4  |
| Setting Up the History Cache and Database .....                                      | 12-6  |
| Flushing the Agent Cache Based on a Time interval .....                              | 12-7  |
| Flushing the Agent Cache Based on the Number of<br>Data Points .....                 | 12-7  |
| Defining How Much Time the Agent Spends Writing Cache<br>to a History File .....     | 12-8  |
| Setting the Number of Days that History is Retained in the<br>History Database ..... | 12-8  |
| Extracting History from the Database .....   | 12-10 |
| Output .....   | 12-10 |
| File and Directory Structure Used by dump_hist .....                                 | 12-11 |
| Before Running the dump_hist Utility .....   | 12-13 |
| Running dump_hist .....  | 12-13 |
| Examples .....   | 12-18 |
| Fixing a Corrupted History Database .....  | 12-20 |

|   |       |
|---|-------|
| Output .....  | 12-21 |
| File and Directory Structure Used by fix_hist ..... | 12-22 |
| Before Running the fix_hist Utility .....           | 12-24 |
| Running fix_hist .....                              | 12-24 |
| Examples .....                                      | 12-27 |

## Chapter 13

### **PATROL Agent Security**

|   |       |
|---|-------|
| PATROL Agent Security Methods .....                       | 13-2  |
| Using PATROL in an Environment with Firewalls .....       | 13-3  |
| Using Access Control List .....                           | 13-4  |
| Using Accounts .....                                      | 13-4  |
| Default Ownership and Permissions for Directories .....   | 13-4  |
| Default Ownership and Permissions for Files .....         | 13-5  |
| Changing Ownership and Permissions on Unix .....          | 13-6  |
| Single Sign-On for Windows 2000 .....                     | 13-8  |
| Supported Platforms, Operating Systems, and Components .. | 13-8  |
| Compatibility .....                                       | 13-9  |
| Security Model .....                                      | 13-9  |
| Location of Files .....                                   | 13-9  |
| Configuring PATROL Agent for Single Sign-On .....         | 13-10 |
| Log Files .....   | 13-10 |
| Configure Rights in Active Directory Domain Trees .....   | 13-11 |
| Disabling Single Sign-On .....                            | 13-11 |

## Chapter 14

### **PATROL Agent and SNMP Concepts**

|  |       |
|--|-------|
| PATROL SNMP Design .....                               | 14-2  |
| Support .....  | 14-2  |
| Architecture .....                                     | 14-3  |
| Dependencies .....                                     | 14-4  |
| Configuration Requirements .....                       | 14-5  |
| Startup Process .....                                  | 14-6  |
| SNMP Roles Available to the PATROL Agent .....         | 14-8  |
| PATROL Agent as SNMP Manager .....                     | 14-8  |
| Variables for Configuring the PATROL Agent .....       | 14-8  |
| PSL Functions for Configuring the PATROL Agent .....   | 14-10 |
| Two Methods of Sending SNMP Traps .....                | 14-11 |
| Comparison of Methods .....                            | 14-12 |
| Diagnostic Information for the PATROL SMUX based SNMP  |       |
| sub-agent .....  | 14-12 |
| Creating Custom Read and Write Community Strings ..... | 14-13 |

|                                      |       |
|--------------------------------------|-------|
| Modifying the snmpagt.cfg file ..... | 14-15 |
| Basic Agent Configuration File ..... | 14-17 |

## Chapter 15

### SNMP Configuration and Implementation Using PEM

|   |       |
|---|-------|
| Before You Begin Configuring PATROL with SNMP .....           | 15-3  |
| Process for Configuring the PATROL Agent with SNMP .....      | 15-3  |
| When Configuration Changes Take Effect .....                  | 15-4  |
| Configuring the PATROL SNMP Master Agent .....                | 15-4  |
| Starting The PATROL SNMP Master Agent .....                   | 15-4  |
| Specify the Working Directory .....                           | 15-8  |
| Starting the PATROL SNMP Master Agent .....                   | 15-8  |
| Configuring the SNMP Subagent and the PATROL Agent .....      | 15-9  |
| Start the PATROL SNMP Subagent Automatically .....            | 15-9  |
| Prevent the PATROL SNMP Master Agent on Unix from             |       |
| Starting Automatically .....                                  | 15-11 |
| Specify the Read Community Name .....                         | 15-11 |
| Specify the Write Community Name .....                        | 15-12 |
| Specify the Port Number for Sending and Receiving Traps ..... | 15-12 |
| Specify the MIB-II Information .....                          | 15-13 |
| Send Traps through PATROL SNMP Master Agent .....             | 15-13 |
| Send Traps Directly to SNMP Managers .....                    | 15-14 |
| Ignoring Return IPAddresses .....                             | 15-14 |
| Configuring the PATROL Event Manager to Send SNMP Traps ..... | 15-15 |
| List of Standard Event Classes .....                          | 15-16 |
| Sending SNMP Traps Based on PATROL Events .....               | 15-20 |
| Destination of SNMP PEM-based Traps .....                     | 15-20 |
| SNMP Support Based on PATROL Event Manager .....              | 15-21 |
| Filter Traps Based on PATROL Event Severity Level .....       | 15-22 |
| Specify Application That Issued the Trap .....                | 15-23 |
| Specify the Node Name Where the Trap Originated .....         | 15-23 |
| Filter Traps Based on PATROL Event ID .....                   | 15-24 |
| Filter Traps Based on PATROL Event Class .....                | 15-24 |
| Filter Events Based on PATROL Event Description .....         | 15-25 |
| Specify a Time Period to Send Traps Based on                  |       |
| PATROL Events .....   | 15-25 |
| Filter Traps Based on PATROL Event Type .....                 | 15-27 |
| Filter Traps Based on PATROL Event Status .....               | 15-28 |
| Set SNMP Trap Format .....                                    | 15-29 |
| Disabling SNMP Trap Support for PATROL Events .....           | 15-31 |

## Chapter 16

### SNMP Configuration and Implementation Using PSL

|  |       |
|--|-------|
| Configuring the PATROL Agent as an SNMP Manager              | 16-2  |
| Listening for SNMP Traps                                     | 16-2  |
| Getting and Setting MIB Variables                            | 16-3  |
| Sending SNMP Traps Based on PSL Functions                    | 16-4  |
| Overview of PSL Functions for SNMP                           | 16-4  |
| Set Default Community Name for PSL SNMP Functions            | 16-5  |
| Managing PSL SNMP Built-In Functions Execution               | 16-6  |
| PSL SNMP Built-In Functions                                  | 16-8  |
| Sending SNMP Traps   | 16-9  |
| Debugging PSL Functions for SNMP                             | 16-9  |
| Interpreting Error Messages from PSL Functions               | 16-10 |
| Distinguishing Instance State Changes from Propagated States | 16-11 |
| Gathering PATROL Data from PATROL MIB Tables                 | 16-12 |

## Chapter 17

### Managing Parameter Overrides

|  |       |
|--|-------|
| Parameter Properties that Can Be Overridden                          | 17-2  |
| Methods of Overriding Parameters                                     | 17-2  |
| External File Overrides  | 17-3  |
| Operator Overrides   | 17-3  |
| PSL Overrides  | 17-3  |
| Developer Overrides  | 17-3  |
| Controlling Which Override Methods Are Used                          | 17-4  |
| If External File Overrides Are Disabled                              | 17-5  |
| If Operator Overrides Are Disabled                                   | 17-5  |
| If PSL Overrides Are Disabled  | 17-5  |
| Combining Multiple Methods of Overriding Parameters                  | 17-6  |
| Using Developer Overrides  | 17-6  |
| Using External File Overrides and Operator Overrides                 | 17-6  |
| Using External File Overrides and PSL Overrides                      | 17-7  |
| Using Operator Overrides and PSL Overrides                           | 17-7  |
| Using External File Overrides, Operator Overrides, and PSL Overrides | 17-7  |
| Using External Override Files to Manage Parameter Overrides          | 17-8  |
| Storing Overrides in a Single File                                   | 17-8  |
| Storing Overrides in Multiple Files                                  | 17-9  |
| External Override File Location and Privileges                       | 17-11 |
| External Override File Names   | 17-11 |
| Defining Which Override Files to Use                                 | 17-12 |
| Using Macro Variables in Override Files                              | 17-13 |

|   |       |
|---|-------|
| External Override File Poll Cycle .....   | 17-13 |
| External Override File Format .....       | 17-14 |
| Deleting the External Override File ..... | 17-17 |
| Additional Functions .....                | 17-17 |

|                   |   |
|-------------------|---|
| <b>Appendix A</b> | <b>List of PATROL Agent Variables</b>   |
| <b>Appendix B</b> | <b>Changing Variables in a File Manually</b>                                    |
|                   | Editing the Change File Using a Text Editor .....                               |
|                   | B-4   |
| <b>Appendix C</b> | <b>Configuring a DCOM Interface to the PATROL Agent</b>                         |
|                   | Setting PATROL Agent Configuration Variables .....                              |
|                   | C-3   |
|                   | Configuring DCOM on PATROL Agents .....   |
|                   | C-4   |
|                   | Configuring Your Computer for DCOM Communication with<br>the PATROL Agent ..... |
|                   | C-6   |
| <b>Appendix D</b> | <b>PATROL SNMP MIB</b>  |
| <b>Appendix E</b> | <b>Power Management Support</b>   |
| <b>Appendix F</b> | <b>Support for Regular Expressions</b>  |
| <b>Appendix G</b> | <b>Managing Unix Environment Variables</b>                                      |
| <b>Appendix H</b> | <b>Managing Performance</b>   |
| <b>Glossary</b>   |   |
| <b>Index</b>      |   |

---

## Tables

|            |  |       |
|------------|--|-------|
| Table 1-1  | Patrol Components . . . . .  | 1-6   |
| Table 1-2  | PATROL Agent File Usage . . . . .  | 1-12  |
| Table 1-3  | Methods to Request Event Management Services . . . . .                   | 1-17  |
| Table 2-1  | Command Line Arguments for Starting PATROL Agent . . . . .               | 2-4   |
| Table 2-2  | Service Connection Point attributes set by the<br>PATROL Agent . . . . . | 2-23  |
| Table 3-1  | Variables That Cannot be Changed . . . . .                               | 3-7   |
| Table 3-2  | Methods for Specifying Files to Use for Changing Variables . . . . .     | 3-10  |
| Table 3-3  | Options for Changing the Values of Agent Variables . . . . .             | 3-11  |
| Table 3-4  | Methods for Changing Variables . . . . .                                 | 3-11  |
| Table 5-1  | Access Control List Connection Modes . . . . .                           | 5-11  |
| Table 5-2  | Connection Modes and Accounts Used by the Agent's Clients . . . . .      | 5-15  |
| Table 6-1  | PATROL Agent Behavior in a Cluster Environment . . . . .                 | 6-7   |
| Table 6-2  | Cluster Administration Properties . . . . .                              | 6-20  |
| Table 6-3  | PATROL Cluster-Specific Environment Variables . . . . .                  | 6-28  |
| Table 6-4  | Operation of Configuration and History Environment Variables . . . . .   | 6-29  |
| Table 7-1  | PATROL Knowledge Module Version Arbitration . . . . .                    | 7-4   |
| Table 7-2  | Knowledge Module Statuses in the PATROL Agent . . . . .                  | 7-6   |
| Table 8-1  | pconfig Command Strings . . . . .  | 8-6   |
| Table 8-2  | pconfig Options . . . . .  | 8-9   |
| Table 9-1  | Menus and Menu Commands for the xpcnfig Utility . . . . .                | 9-5   |
| Table 9-2  | Options for Starting the xpcnfig from the Command Line . . . . .         | 9-9   |
| Table 9-3  | Tasks to Handle Variables from the Primary Window . . . . .              | 9-32  |
| Table 9-4  | Handling Variables from the Edit Variable Dialog Box . . . . .           | 9-41  |
| Table 10-1 | Menus and Menu Commands for the wpcnfig Utility . . . . .                | 10-4  |
| Table 11-1 | Information Stored in Event Log . . . . .                                | 11-3  |
| Table 11-2 | Location of Event Log Files . . . . .                                    | 11-4  |
| Table 11-3 | dump_events Utility Directory Structure . . . . .                        | 11-10 |

|             |   |       |
|-------------|---|-------|
| Table 11-4  | PATROL_HOME Environment Variable . . . . .                              | 11-11 |
| Table 11-5  | Options for the dump_events Command Line Utility . . . . .              | 11-12 |
| Table 11-6  | Location of Event Archive Files. . . . .                                | 11-15 |
| Table 11-7  | Information Stored in PATROL Error Log. . . . .                         | 11-16 |
| Table 11-8  | Location of PATROL Agent Error Log File . . . . .                       | 11-17 |
| Table 11-9  | Type of Information Recorded in the Audit Log . . . . .                 | 11-22 |
| Table 11-10 | AgentSetup/auditLog Keys and Values . . . . .                           | 11-24 |
| Table 11-11 | Audit Log File Format . . . . .   | 11-25 |
| Table 12-1  | Information Stored in History Database. . . . .                         | 12-3  |
| Table 12-2  | Location of History Files . . . . .                                     | 12-5  |
| Table 12-3  | dump_hist Directories . . . . .   | 12-11 |
| Table 12-4  | Files Read by dump_hist . . . . .                                       | 12-12 |
| Table 12-5  | PATROL_HOME Environment Variable . . . . .                              | 12-13 |
| Table 12-6  | dump_hist Utility Options . . . . .                                     | 12-15 |
| Table 12-7  | Options for -format of dump_hist. . . . .                               | 12-16 |
| Table 12-8  | Directories for fix_hist . . . . .                                      | 12-22 |
| Table 12-9  | Files Read by fix_hist. . . . .   | 12-23 |
| Table 12-10 | Options Recognized by the fix_hist Utility . . . . .                    | 12-25 |
| Table 13-1  | Methods for Maintaining Security for the PATROL Agent . . . .           | 13-2  |
| Table 13-2  | Directories for Ownership and Permissions of Agent Log . . . .          | 13-4  |
| Table 13-3  | Default Owner and Permissions of Log and Files . . . . .                | 13-5  |
| Table 13-4  | Single Sign-On Security Supported Platforms<br>and Components . . . . . | 13-8  |
| Table 14-1  | PATROL SNMP Component Definitions . . . . .                             | 14-4  |
| Table 14-2  | Configuration for PATROL Support of SNMP. . . . .                       | 14-5  |
| Table 14-3  | Configuring the PATROL Agent to Run with SNMP. . . . .                  | 14-9  |
| Table 14-4  | Functions for Sending Traps. . . . .                                    | 14-10 |
| Table 14-5  | Functions for Starting and Stopping the SNMP Agent . . . . .            | 14-10 |
| Table 14-6  | Functions for Changing the Registered SNMP Manager List . . . .         | 14-11 |
| Table 14-7  | Comparing Methods for Sending Traps . . . . .                           | 14-12 |
| Table 15-1  | Standard Event Classes for Sending SNMP Traps. . . . .                  | 15-16 |
| Table 16-1  | Functions for SNMP Manager/Trap Handler . . . . .                       | 16-2  |
| Table 16-2  | Functions for Getting and Setting MIB Variables . . . . .               | 16-3  |
| Table 16-3  | PSL Functions for SNMP Support. . . . .                                 | 16-8  |
| Table 16-4  | Functions for Sending Traps. . . . .                                    | 16-9  |
| Table 16-5  | Functions for Debugging PSL Functions . . . . .                         | 16-9  |
| Table 16-6  | Global Error Messages for SNMP PSL Functions. . . . .                   | 16-10 |
| Table 16-7  | “status” and “ruleState” Variables . . . . .                            | 16-11 |
| Table 17-1  | Attribute and Value Pairs . . . . .                                     | 17-14 |
| Table A-1   | /AgentSetup/ Variables. . . . .   | A-2   |

|           |  |      |
|-----------|--|------|
| Table A-2 | /AgentSetup/AgentTuning Variables . . . . .                          | A-8  |
| Table A-3 | /AgentSetup/security Variables . . . . .                             | A-9  |
| Table A-4 | /SNMP Variables . . . . .  | A-10 |
| Table A-5 | /AgentSetup/XPC Variables. . . . .                                   | A-13 |
| Table A-6 | Environment Variables Used by PATROL Console and PATROL<br>AgentA-15 |      |
| Table C-1 | PATROL Agent COM Variables . . . . .                                 | C-3  |
| Table F-1 | Wildcards Supported by PATROL Regular Expressions . . . . .          | F-4  |



## Figures

|             |   |       |
|-------------|---|-------|
| Figure 1-3  | <i>PATROL_HOME</i> Directories and Files . . . . .              | 1-11  |
| Figure 2-1  | The PATROL Agent Configuration Dialog Box . . . . .             | 2-18  |
| Figure 2-2  | Windows NT Services Applet Dialog Box . . . . .                 | 2-20  |
| Figure 2-3  | Windows NT Services Applet Dialog Box . . . . .                 | 2-25  |
| Figure 3-1  | Configuring Agent Variables with the Default and Change File    | 3-2   |
| Figure 3-2  | Getting Configuration Information from a PATROL Agent . . . . . | 3-3   |
| Figure 3-3  | Getting Configuration Information from Another Change File .    | 3-3   |
| Figure 3-4  | Purging the PATROL Agent Configuration . . . . .                | 3-4   |
| Figure 3-5  | Applying a Change File to an Agent . . . . .                    | 3-4   |
| Figure 3-6  | Reloading an Agent's Default Configuration . . . . .            | 3-5   |
| Figure 3-7  | Process Flow for Configuring the PATROL Agent . . . . .         | 3-9   |
| Figure 9-1  | xpconfig Primary Window . . . . .                               | 9-7   |
| Figure 9-2  | The Restart Agent Dialog Box . . . . .                          | 9-13  |
| Figure 9-3  | The Reload Configuration Dialog Box . . . . .                   | 9-15  |
| Figure 9-4  | The Kill Agent Dialog Box . . . . .                             | 9-17  |
| Figure 9-5  | The Configuration File Dialog Box . . . . .                     | 9-20  |
| Figure 9-6  | The Purge Configuration Dialog Box . . . . .                    | 9-29  |
| Figure 9-7  | The File Selection Dialog Box . . . . .                         | 9-31  |
| Figure 9-8  | The Add Hosts Dialog Box . . . . .                              | 9-34  |
| Figure 9-9  | The Add Variable Dialog Box . . . . .                           | 9-36  |
| Figure 9-10 | The Edit Variable Dialog Box . . . . .                          | 9-42  |
| Figure 10-2 | wpconfig Toolbar . . . . .                                      | 10-6  |
| Figure 10-3 | The Get Configuration Dialog Box . . . . .                      | 10-10 |
| Figure 10-4 | The Add Variable Dialog Box . . . . .                           | 10-12 |
| Figure 10-5 | The Modify Variable Dialog Box . . . . .                        | 10-16 |
| Figure 10-6 | Toolbar in the Modify Variable Dialog Box . . . . .             | 10-16 |
| Figure 10-7 | The Change Entry Dialog Box . . . . .                           | 10-17 |
| Figure 10-8 | The Modify Variable Dialog Box . . . . .                        | 10-19 |

|              |   |       |
|--------------|---|-------|
| Figure 10-9  | The Set Default Account Dialog Box                        | 10-19 |
| Figure 10-10 | The View Changes Dialog Box                               | 10-24 |
| Figure 10-11 | The Apply Configuration Dialog Box                        | 10-26 |
| Figure 11-1  | Adding /AgentSetup/auditLog in wpconfig\xpconfig          | 11-23 |
| Figure 14-1  | PATROL SNMP Master Agent Interacts with Other Agents      | 14-3  |
| Figure 15-1  | Configuring the PATROL Agent with SNMP                    | 15-3  |
| Figure B-1   | PATROL Agent Configuration Change File Format             | B-3   |
| Figure C-1   | PATROL Client Configuration Dialog Box - ESI Library Tab  | C-7   |
| Figure C-2   | PATROL Client Configuration Dialog Box - Type Library Tab | C-7   |

---

## About This Book

This book contains detailed information about PATROL<sup>®</sup> Agent and is intended for PATROL system administrators. This book explains how the PATROL Agent interacts with other PATROL components, and gives getting started and configuration information. It also describes configuration utilities and Management Information Base (MIB) utilities used with the PATROL Agent.

---

### Note

---

This book assumes that you are familiar with your host operating system. You should know how to perform basic actions in a window environment, such as choosing menu commands and dragging and dropping icons.

---

## How This Book Is Organized

This book is organized as follows. In addition, a glossary of terms and an index appear at the end of the book.

| Chapter/Appendix | Title                                    | Description   |
|------------------|--|---|
| 1                | “Structure of PATROL”                    | provides an overview of the features and components of PATROL |
| 2                | “Starting and Stopping the PATROL Agent” | describes how to start the agent                              |

| <b>Chapter/Appendix</b> | <b>Title</b>   | <b>Description</b>  |
|-------------------------|--|---|
| 3                       | “Background About Configuring the PATROL Agent”                      | provides an overview on controlling the configuration of the agent, the process of configuration, and the methods available for configuration |
| 4                       | Chapter 4, “Establishing Accounts and Ports”                         | describes how to set up accounts and ports and specify which applications and instances use which accounts                                    |
| 5                       | Chapter 5, “Managing Console Connections”                            | describes how to manage the PATROL Agent’s relationship with various consoles and utilities.  |
| 6                       | Chapter 6, “Support of Clusters and Failovers”                       | discusses how the PATROL Agent supports an application in cluster environment and what type of failover tolerance it provides                 |
| 7                       | Chapter 7, “Loading and Monitoring Applications”                     | describes how to manage when, if, and how the PATROL Agent monitors an application  |
| 8                       | “Using pconfig to Configure the PATROL Agent”                        | describes how to configure the agent from the command line  |
| 9                       | “Using xpconfig (Unix) to Configure the PATROL Agent”                | describes how to configure the agent using the xpconfig GUI for Unix  |
| 10                      | Chapter 10, “Using wpconfig (Windows) to Configure the PATROL Agent” | describes how to configure the agent using the wpconfig GUI for Windows   |
| 11                      | “PATROL Agent Logs”  | describes the format, location, and type of history data, event data, and log data stored by the PATROL Agent                                 |
| 12                      | “PATROL Agent Parameter History”                                     | describes three agent command line utilities which allow you to dump parameter history data, event data, and repair corrupted history data    |
| 13                      | “PATROL Agent Security”  | describes methods for maintaining security for the agent  |
| 14                      | Chapter 14, “PATROL Agent and SNMP Concepts”                         | describes how the PATROL Agent and PATROL SNMP Master Agent interact and what type of information the agents gather and send                  |
| 15                      | Chapter 15, “SNMP Configuration and Implementation Using PEM”        | describes how to use PATROL’s built-in SNMP support to send traps based on PATROL events  |

| <b>Chapter/Appendix</b> | <b>Title</b>  | <b>Description</b>  |
|-------------------------|---|---|
| 16                      | Chapter 16, "SNMP Configuration and Implementation Using PSL" | provides an overview of PSL support for SNMP  |
| 17                      | Chapter 17, "Managing Parameter Overrides"                    | describes multiple methods for implementing parameter overrides   |
| A                       | Chapter , "List of PATROL Agent Variables"                    | lists each PATROL Agent variable, describes each's purpose, and points to where you can find additional information               |
| B                       | "Changing Variables in a File Manually"                       | describes how to change variables manually  |
| C                       | "Configuring a DCOM Interface to the PATROL Agent"            | describes how to set up the PATROL Agent for use with the Distributed Component Object Model (DCOM) programming interface         |
| D                       | "PATROL SNMP MIB"   | describes the MIB tables contained in \$PATROL_HOME/lib/patrol.mib  |
| E                       | "Power Management Support"                                    | describes the how the PATROL Agent supports OnNow and ACPI power management   |
| F                       | "Support for Regular Expressions"                             | describe metacharacters used for regular expressions  |
| G                       | "Managing Unix Environment Variables"                         | describes how to store Unix environment variables in a shell script file that will not be overwritten by subsequent installations |
| H                       | "Managing Performance"  | describes how to alter some of the PATROL Agent processes to achieve better performance.  |
| Glossary                | "Glossary"  | lists PATROL specific terminology   |
| Index                   | "Index"   | lists index entries   |

# Related Documentation

BMC Software products offer several types of documentation:

- online and printed books
- online Help
- release notes

In addition to this book and the online Help, you can find useful information in the publications listed in the following table. As “Online and Printed Books” on page -xxvii explains, these publications are available on request from BMC Software.

| <b>Category</b>        | <b>Document</b>                                    | <b>Description</b>  |
|------------------------|--|---|
| installation documents | <i>PATROL for Windows Installation Guide</i>       | contains the installation instructions for PATROL Console for Windows, PATROL Agent for Windows, and Knowledge Modules. |
|                        | <i>PATROL for Unix Installation Guide</i>          | contains the installation instructions for PATROL Console for Unix, PATROL Agent for Unix, and Knowledge Modules.       |
|                        | <i>PATROL for OpenVMS Installation Guide</i>       | contains the installation instructions for PATROL Agent for OpenVMS and Knowledge Modules.                              |
| console documents      | <i>PATROL for Windows User Guide</i>               | provides instructions on how to use the PATROL Console for Windows to customize PATROL object classes and attributes    |
|                        | <i>PATROL Console for Unix User Guide</i>          | provides instructions on how to use the PATROL Console for Unix to customize PATROL object classes and attributes       |
| supplemental documents | <i>PATROL Agent for Windows 2000 Release Notes</i> | contains the latest updates to PATROL Agent for Windows   |
|                        | <i>PATROL Agent for Unix Release Notes</i>         | contains the latest updates to PATROL Agent for Unix  |

## Online and Printed Books

The books that accompany BMC Software products are available in online format and printed format. You can view online books with Acrobat Reader from Adobe Systems. The reader is provided at no cost, as explained in “To Access Online Books.” You can also obtain additional printed books from BMC Software, as explained in “To Request Additional Printed Books.”

### To Access Online Books

Online books are formatted as Portable Document Format (PDF) files. You can view them, print them, or copy them to your computer by using Acrobat Reader 3.0 or later. You can access online books from the documentation compact disc (CD) that accompanies your product or from the World Wide Web.

In some cases, installation of Acrobat Reader and downloading the online books is an optional part of the product installation process. For information about downloading the free reader from the Web, go to the Adobe Systems site at <http://www.adobe.com>.

To view any online book that BMC Software offers, visit the support page of the BMC Software Web site at <http://www.bmc.com/support.html>. Log on and select a product to access the related documentation. (To log on, first-time users can request a user name and password by registering at the support page or by contacting a BMC Software sales representative.)

### To Request Additional Printed Books

BMC Software provides a core set of printed books with your product order. To request additional books, go to <http://www.bmc.com/support.html>.

## Online Help

You can access Help for a product through the product's Help menu. The online Help provides information about the product's graphical user interface (GUI) and provides instructions for completing tasks.

## Release Notes

Printed release notes accompany each BMC Software product. Release notes provide up-to-date information such as

- updates to the installation instructions
- last-minute product information

The latest versions of the release notes are also available on the Web at **<http://www.bmc.com/support>**.

# Conventions

The following conventions are used in this book:

- This book includes special elements called *notes*, *warnings*, *examples*, and *tips*:

---

## Note

---

Notes provide additional information about the current subject.

---

---

## Warning

---

Warnings alert you to situations that can cause problems, such as loss of data, if you do not follow instructions carefully.

---

---

## Example

---

An example clarifies a concept discussed in text.

---

---

## Tip

---

A tip provides useful information that may improve product performance or make procedures easier to follow.

---

- All syntax, operating system terms, and literal examples are presented in this typeface.
- In instructions, **boldface** type highlights information that you enter. File names, directories, and Web addresses also appear in boldface type.
- The symbol => connects items in a menu sequence. For example, **Actions => Create Test** instructs you to choose the Create Test command from the Actions menu.
- The symbol >> denotes one-step instructions.

- In syntax, path names, or system messages, *italic* text represents a variable, as shown in the following examples:

The table *table\_name* is not available.

***system/instance/file\_name***

- In syntax, the following additional conventions apply:
  - A vertical bar ( | ) separating items indicates that you must choose one item. In the following example, you would choose *a*, *b*, or *c*:  
  
a | b | c
  - An ellipsis ( . . . ) indicates that you can repeat the preceding item or items as many times as necessary.
  - Square brackets ( [ ] ) around an item indicate that the item is optional.
- The following table shows equivalent mouse buttons for Unix users and Windows users:

| <b>Unix Button</b> | <b>Windows Button</b> | <b>Description</b>   |
|--------------------|-----------------------|--|
| MB1                | left mouse button     | Click this button on an icon or menu command to select that icon or command. Click MB1 on a command button to initiate action. Double-click an icon to open its container. |
| MB2                | not applicable        | Click this button on an icon to display the InfoBox for the icon. To simulate MB2 on a two-button mouse, simultaneously press the two buttons (MB1 and MB3).               |
| MB3                | right mouse button    | Click this button on an icon to display its pop-up menu.   |

---

**Note**

---

If you have a one-button mouse (such as an Apple Macintosh mouse), assign MB1 to that button. You should also define a user-selectable combination of option and arrow keys to simulate MB2 and MB3. For details, refer to the documentation for your emulation software.

---



---

# Structure of PATROL

This chapter lists the components of PATROL and provides an overview of the PATROL architecture. It also gives background information about the PATROL Event Manager Engine (the component of the PATROL Agent that manages events), lists the services available from the PATROL Agent, and tells you how to request services from the PATROL Agent.

This chapter contains the following sections:

|   |      |
|---|------|
| Role of the PATROL Agent . . . . .                  | 1-2  |
| Purpose of PATROL . . . . .                         | 1-2  |
| Overview of the PATROL 3.x Architecture . . . . .   | 1-3  |
| Components of PATROL . . . . .                      | 1-6  |
| PATROL Components . . . . .                         | 1-6  |
| Non-PATROL Components . . . . .                     | 1-9  |
| PATROL Agent Directory and File Structure . . . . . | 1-10 |
| PATROL_HOME Directory Structure . . . . .           | 1-10 |
| Files Used by the PATROL Agent . . . . .            | 1-11 |
| Event Management with the PATROL Agent . . . . .    | 1-14 |
| Services Provided by the PATROL Agent . . . . .     | 1-14 |
| Overview of the PEM Engine . . . . .                | 1-16 |
| Management Services from the PEM Engine . . . . .   | 1-17 |

# Role of the PATROL Agent

This section describes how the PATROL Agent works with a number of different PATROL and non-PATROL system components.

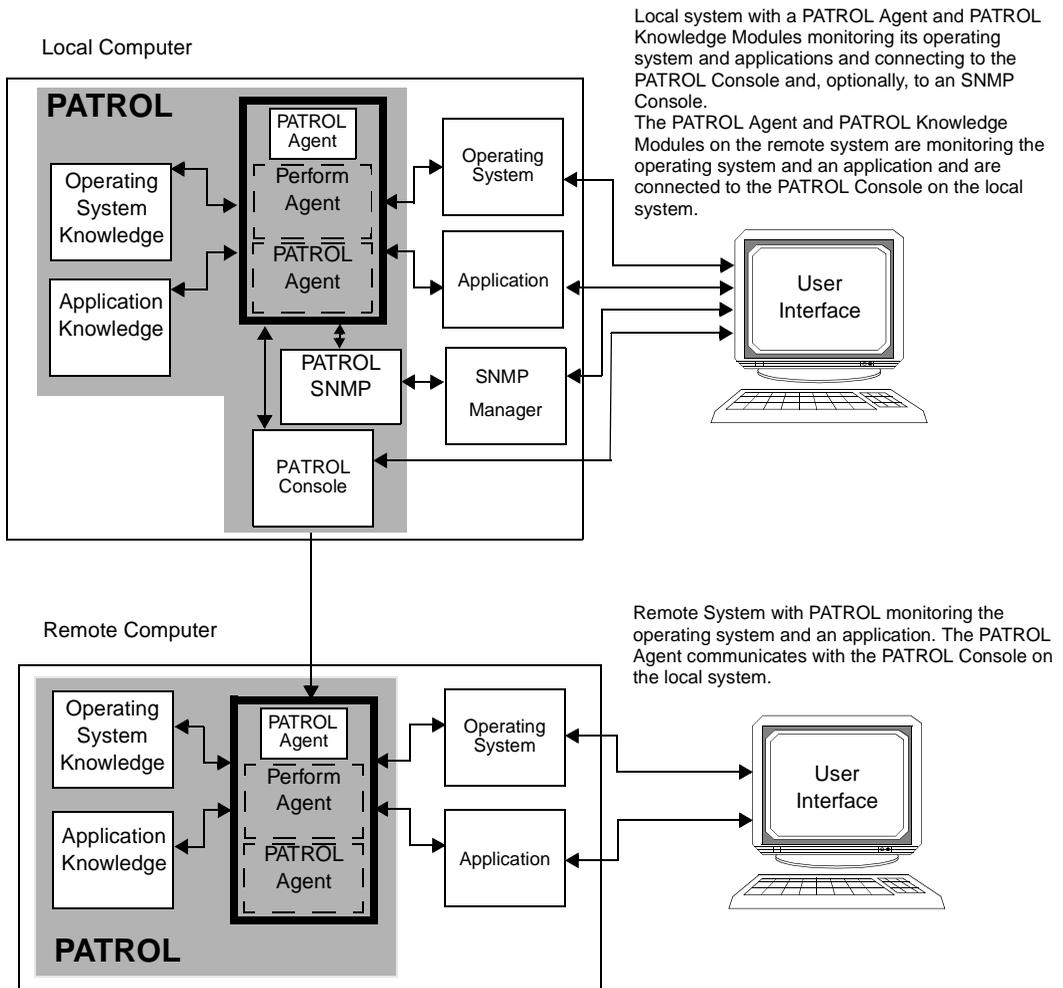
## Purpose of PATROL

PATROL is a systems, applications, and event management tool. It provides an environment where you can monitor the status of every vital resource in the distributed environment you are managing.

# Overview of the PATROL 3.x Architecture

Figure 1-1 shows the relationship between the PATROL Agent and other PATROL and non-PATROL components in a PATROL 3.x environment. The agent's interactions with these components is described in the following sections.

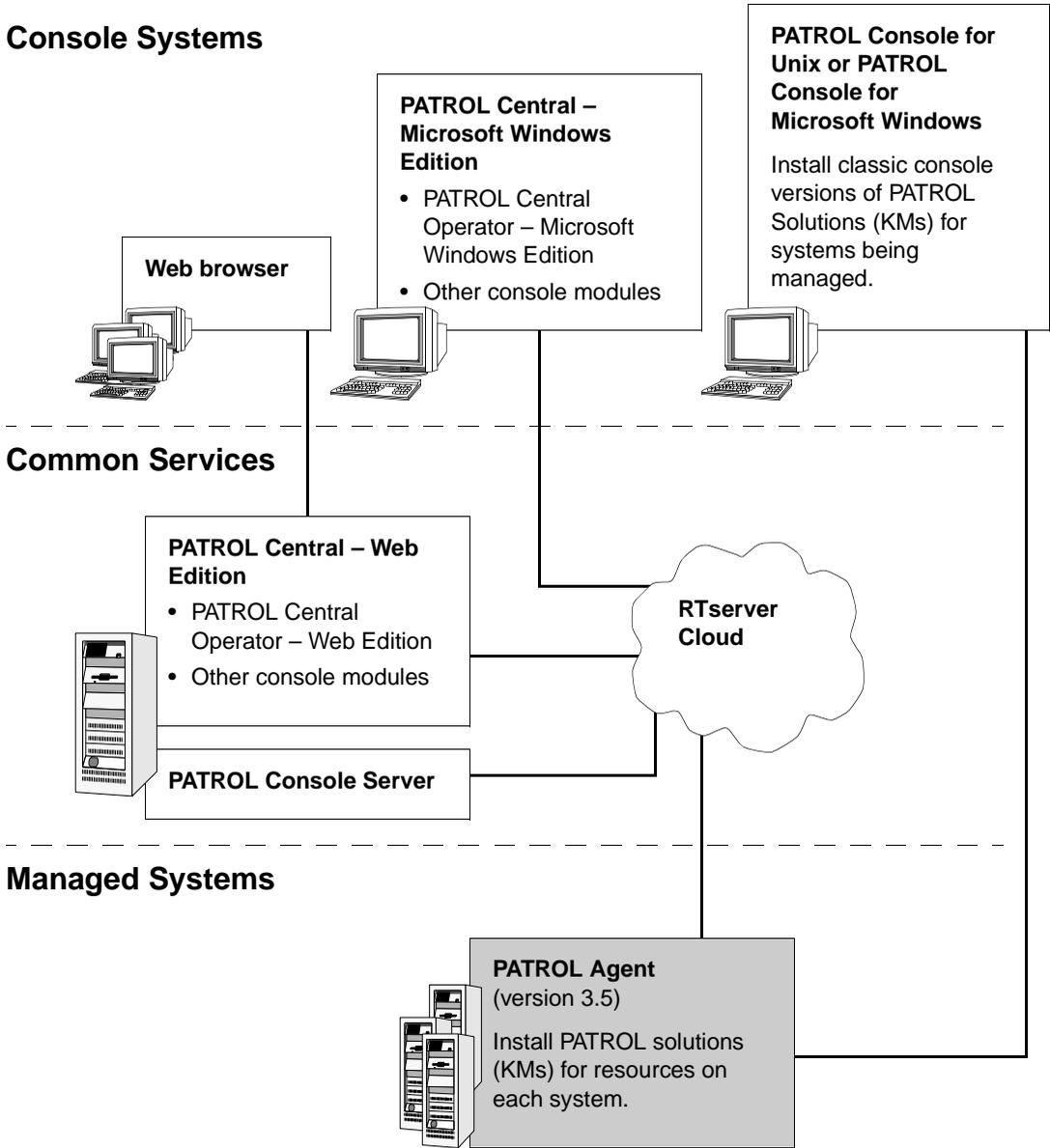
**Figure 1-1 The PATROL Agent in a 3.x PATROL System**



## Overview of the PATROL 7.x Architecture

Figure 1-2 shows the relationship between the PATROL Agent and other PATROL components in a PATROL 7.x environment. The difference is that the agent talks to 7.x consoles through the PATROL Console Server and an RTserver or RTserver cloud.

Figure 1-2 The PATROL Agent in a 7.x PATROL System



PATROL products and solutions may require additional files installed throughout the infrastructure.

# Components of PATROL

This section lists and describes the components provided as part of PATROL and provides an overview of the products from other vendors that you can integrate with PATROL.

## PATROL Components

**Table 1-1** Patrol Components (Part 1 of 4)

| <b>Component</b>                                    | <b>Purpose</b>  | <b>Dependencies</b>   |
|---|---|---|
| PATROL Central Operator - Microsoft Windows Edition | Like other PATROL Consoles, PATROL Central Operator - Microsoft Windows Edition provides a window into your PATROL environment. PATROL Central Operator - Microsoft Windows Edition works with only the PATROL 7.x architecture; it does not work with the PATROL 3 architecture.   | <ul style="list-style-type: none"><li>• PATROL Console Server</li><li>• PATROL Agent</li></ul>    |
| PATROL Central Operator - Web Edition               | PATROL Central Operator - Web Edition is a web-based application that provides a window into your PATROL environment. PATROL Central Operator - Web Edition works with only the PATROL 7.x architecture; it does not work with the PATROL 3 architecture.   | <ul style="list-style-type: none"><li>• PATROL Console Server</li><li>• PATROL Agent</li></ul>    |
| PATROL Console Server                               | The PATROL Console Server is used by PATROL 7.x consoles to communicate with PATROL Agents on managed systems. A primary feature of PATROL monitoring is that you can create views and organize objects according to a specific business need. The PATROL Console Server provides persistency support for those views. In addition, persistent views are maintained in a common format so you can use the same view with multiple 7.x consoles. | <ul style="list-style-type: none"><li>• PATROL Agent</li><li>• PATROL Knowledge Modules</li></ul> |

**Table 1-1 Patrol Components (Part 2 of 4)**

| Component                                 | Purpose   | Dependencies  |
|---|---|---|
| <p>PATROL Operator Console</p>            | <p>The graphical work space from which you issue commands and manage the distributed environment monitored by PATROL. The PATROL Console displays all of the monitored computers and applications.</p> <p>With the PATROL Operator Console you can perform these tasks:</p> <ul style="list-style-type: none"> <li>• define which applications you want PATROL to monitor</li> <li>• monitor and manage computers and applications through the PATROL Agent and PATROL Knowledge Modules</li> <li>• monitor the PATROL Agent's use of resources</li> <li>• run predefined or user-defined commands and tasks against monitored machines</li> <li>• run state change action commands on the PATROL Console machine when a state change occurs on a monitored computer</li> <li>• log on to any managed computer (only for Unix and OpenVMS.)</li> <li>• start and stop PATROL Agents remotely</li> <li>• view parameter data</li> <li>• retrieve historical data stored by the PATROL Agent</li> </ul> | <ul style="list-style-type: none"> <li>• PATROL Agent</li> <li>• PATROL Knowledge Module</li> </ul> |
| <p>PATROL Developer Console</p>           | <p>With the PATROL Developer Console, you can do everything you can do with the PATROL Operator Console, plus the following:</p> <ul style="list-style-type: none"> <li>• build new KMs and customize existing KMs</li> <li>• customize menu commands and application parameters</li> <li>• modify agent knowledge in memory</li> <li>• transfer knowledge to an agent</li> <li>• edit or replace KM files</li> <li>• send an additional KM file to an agent machine</li> <li>• start the PATROL Agent configuration utility</li> <li>• commit changes to agents</li> </ul>   | <ul style="list-style-type: none"> <li>• PATROL Agent</li> <li>• PATROL Knowledge Module</li> </ul> |
| <p>PATROL Event Manager (PEM Console)</p> | <p>With the PEM Console, you can do the following:</p> <ul style="list-style-type: none"> <li>• view events</li> <li>• manage events and use events to control your environment</li> <li>• trigger events</li> <li>• generate event statistics</li> <li>• acknowledge events</li> <li>• delete events</li> <li>• close events</li> </ul>  | <ul style="list-style-type: none"> <li>• PATROL Agent</li> <li>• PATROL Knowledge Module</li> </ul> |

**Table 1-1 Patrol Components (Part 3 of 4)**

| Component                   | Purpose  | Dependencies  |
|-----------------------------|--|---|
| PATROL Integration Products | Provides an interface between PATROL Agent and third party enterprise management systems such as HP OpenView Network Node Manager.   | <ul style="list-style-type: none"> <li>• PATROL Agent</li> <li>• PATROL Knowledge Module</li> </ul>                           |
| PATROL Agent                | <p>The core piece of the PATROL architecture that monitors and manages host computers. The PATROL Agent consists of Perform Agent executables (BDS_SDService.exe, which is registered as a service on Windows platforms, and bgscollect.exe) and the PATROL Agent executable (patrolagent.exe), which is registered as a service. On the OpenVMS platform, the Agent does not use the Perform Agent executables.</p> <p>The PATROL Agent performs the following tasks:</p> <ul style="list-style-type: none"> <li>• runs commands to collect system or application information; the information is collected according to applications and parameters defined in Knowledge Modules</li> <li>• stores information locally for retrieval by the PATROL Console</li> <li>• loads specified Knowledge Modules (KMs) at start-up, runs menu commands, and updates InfoBoxes on the PATROL Console</li> <li>• acts as a service provider for event management</li> </ul> | <ul style="list-style-type: none"> <li>• PATROL Knowledge Module</li> </ul>   |
| PATROL Knowledge Module     | <p>A set of files from which a PATROL Agent receives information about all of the resources, such as databases and file systems, running on a monitored computer.</p> <p>PATROL KMs provide information to the PATROL Agent about:</p> <ul style="list-style-type: none"> <li>• the identity of objects</li> <li>• parameters</li> <li>• actions to take when an object changes a state</li> <li>• how to monitor the application</li> </ul>   | <ul style="list-style-type: none"> <li>• PATROL Agent</li> </ul>  |
| Agent Query                 | You can view information about monitored objects through Agent Query. Using this function, you can work with objects outside the PATROL Console window. Instead of viewing objects as icons, you view their names in a tabular format. Results of queries are displayed in the Query Results window. You can save queries, then load and reissue them when needed.   | <ul style="list-style-type: none"> <li>• PATROL Agent</li> <li>• PATROL Knowledge Module</li> <li>• PATROL Console</li> </ul> |

**Table 1-1 Patrol Components (Part 4 of 4)**

| <b>Component</b>              | <b>Purpose</b>   | <b>Dependencies</b>   |
|-------------------------------|--|---|
| PATROL Script Language (PSL)  | Included as part of the PATROL Developer Console is the PATROL Script Language (PSL) that you can use to write parameters, commands, tasks, recovery actions, and discovery procedures for the PATROL Agents. PSL is a fourth-generation language that is both compiled and interpreted, which is similar to programming languages such as C++, C, and Perl. PSL is the native language of PATROL KMs.   | <ul style="list-style-type: none"> <li>• PATROL Agent</li> </ul>  |
| PSL Compiler                  | You can use the PSL Compiler to check the syntax of PSL and to build a PSL library.  | <ul style="list-style-type: none"> <li>• PATROL Agent</li> <li>• PATROL Developer Console</li> </ul>                          |
| PSL Debugger                  | You can use the PSL Debugger to debug PSL scripts.   | <ul style="list-style-type: none"> <li>• PATROL Agent</li> <li>• PATROL Developer Console</li> </ul>                          |
| PATROL Command Line Interface | <p>The PATROL Command Line Interface (CLI) is a program for retrieving object and event information from a PATROL Agent. CLI is designed to connect to a PATROL Agent in instances when a GUI interface is unavailable or when the user is logged onto a host using a terminal emulator (without a TCP/IP stack).</p> <p>CLI has both interactive and non-interactive modes. You can start CLI from a command line and manually submit commands to the CLI. You can also call the CLI within a script and have it execute commands that you provide on the command line or in additional CLI script files.</p> | <ul style="list-style-type: none"> <li>• PATROL Agent</li> <li>• PATROL Knowledge Module</li> <li>• PATROL Console</li> </ul> |

## Non-PATROL Components

The PATROL Agent can run with console products (managers) and agents from other vendors. These managers and agents can communicate with the PATROL Agent through SNMP or through the PATROL API. For more information, see Chapter 14, “PATROL Agent and SNMP Concepts,” and “Event Management with the PATROL Agent.”

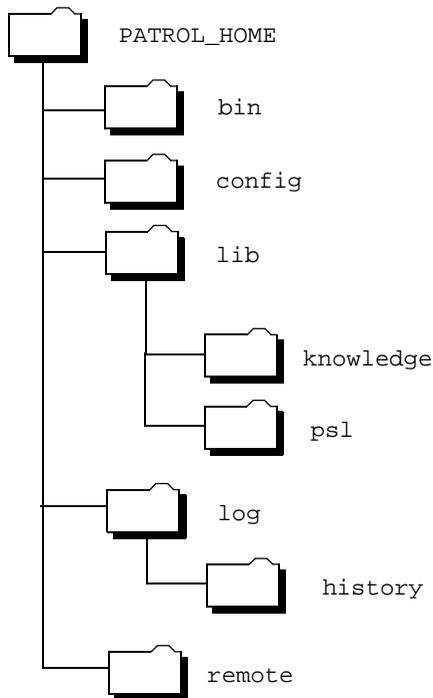
# PATROL Agent Directory and File Structure

The PATROL Agent creates several important directories and files on the PATROL host's disk drive. These files are automatically generated at run time and range from temporary files used for error logging to permanent files that store historical information.

## ***PATROL\_HOME*** Directory Structure

The *PATROL\_HOME* directory is the directory in which the PATROL installation process installs PATROL and in which the PATROL executables and utilities run, write, and read information. The directory contains binaries, knowledge, sounds, images, application defaults, help files, and utilities. The directory is represented as \$PATROL\_HOME in Unix and %PATROL\_HOME% in Windows. Figure 1-3 shows the *PATROL\_HOME* directory structure.

**Figure 1-3** *PATROL\_HOME* Directories and Files



---

**Note**

This brief discussion of PATROL’s directory structure is not exhaustive. The directories depicted in Figure 1-3 and the files listed in Table 1-2 are the primary directories and files in a PATROL installation. In your PATROL implementation, additional files and directories may exist depending upon which platform you run on and which components you install.

---

## Files Used by the PATROL Agent

Table 1-2 lists the types of files used by the PATROL Agent and the directory in which it stores these files.

**Table 1-2 PATROL Agent File Usage (Part 1 of 2)**

| File Type                              | Unix Path   | Description and Examples  |
|--|---|---|
|  | Windows Path  |   |
| executable, shared library             | \$PATROL_HOME/bin   | binary directory  |
|  | %PATROL_HOME%\bin   |   |
| agent configuration file (default)     | \$PATROL_HOME/lib/config.default  | agent startup default configuration file                                      |
|  | %PATROL_HOME%\lib\config.default  |   |
| event catalog                          | \$PATROL_HOME/lib/knowledge/StdEvents.ctg   | Standard Events Catalog   |
|  | %PATROL_HOME%\lib\knowledge\StdEvents.ctg   |   |
| message catalogs                       | \$PATROL_HOME/lib/nls/C/1   | directory of message catalogs   |
|  | %PATROL_HOME%\lib\nls\C\1   |   |
| PSL and OS scripts (shipped)           | \$PATROL_HOME/lib/psl   | directory for shipped PSL and OS (and other KM-defined command types) scripts |
|  | %PATROL_HOME%\lib\psl   |   |
| KMs and Standard Event Catalogs        | \$PATROL_HOME/lib/knowledge   | directory for shipped KMs and Standard Events Catalog                         |
|  | %PATROL_HOME%\lib\knowledge   |   |
| commit lock                            | \$PATROL_HOME/lib/commit.lock   | file created temporarily during a Commit operation                            |
|  | %PATROL_HOME%\lib\commit.lock   |   |
| agent configuration files (customized) | \$PATROL_HOME/config/config_<hostname>-<portnumber>                               | agent configuration files   |
|  | %PATROL_HOME%\config\config_<hostname>-<portnumber>                               |   |
| history file and indices               | \$PATROL_HOME/log/history/<hostname>/<portnumber>/ annotate.dat, dir, param.hist  | history file and indices  |
|  | %PATROL_HOME%\log\history\<hostname>\<portnumber>>\ annotate.dat, dir, param.hist |   |
| PEM event log                          | \$PATROL_HOME/log/PEM_<hostname>_<portnumber>.log, archive                        | files containing all events for the PATROL Event Manager                      |
|  | %PATROL_HOME%\log\PEM_<hostname>_<portnumber>.log, archive                        |   |

**Table 1-2 PATROL Agent File Usage (Part 2 of 2)**

| <b>File Type</b>        | <b>Unix Path</b>   | <b>Description and Examples</b>                  |
|-------------------------|--|--|
|                         | <b>Windows Path</b>  |  |
| PEM event lock          | <i>/\$PATROL_HOME/log/PEM_&lt;hostname&gt;_&lt;portnumber&gt;.log-lock</i>                   | lock on an Event Manager file                    |
|                         | <i>%PATROL_HOME%\log\PEM_&lt;host name&gt;_&lt;portnumber&gt;.log-lock</i>                   |  |
| Error log for the Agent | <i>\$PATROL_HOME/log/PatrolAgent-&lt;hostname&gt;-&lt;portnumber&gt;.errs.~&lt;iter&gt;~</i> | file containing agent error messages, and so on. |
|                         | <i>%PATROL_HOME%\log\PatrolAgent-&lt;hostname&gt;-&lt;portnumber&gt;.errs.~&lt;iter&gt;~</i> |  |

# Event Management with the PATROL Agent

The PATROL Agent uses its PATROL Event Manager (PEM) engine for event management. You can access an agent's PEM Engine in the following ways:

- from the PATROL Console, PEM Console for Unix, or any of the PATROL Integration products
- from any PATROL Knowledge Module loaded on the agent by using the PATROL Script Language (PSL)
- from a C program, using the PATROL API
- from the command line, using the PATROL Command Line Interface

## Services Provided by the PATROL Agent

Through the PEM Engine, the PATROL Agent provides the following event management services:

- view, filter, and sort events
- define and access event knowledge through event catalogs

An event catalog is a subset of a PATROL Knowledge Module.

- trigger events from PSL, a C program, the command line, or a PEM Console
- send ActiveX scripts to the PATROL Scripting Host to run (for Windows only) with the PSL `execute` function

For more information on using the PATROL Scripting Host, refer to the *PATROL Script Language Reference Guide*.

- configure commands (standard or user created) for execution by the agent when an event is triggered

When the event is triggered, the agent checks the event catalog to determine whether a Notification command is defined. If it is, the agent executes the command.

- schedule commands (standard or user created) for execution by the agent when an event is escalated

When the event is escalated, the agent checks the event catalog to determine whether an Escalation command is defined. If it is, the agent executes the command.

- store events, even when no console is registered to receive them
- manage events through `acknowledge/close/delete`

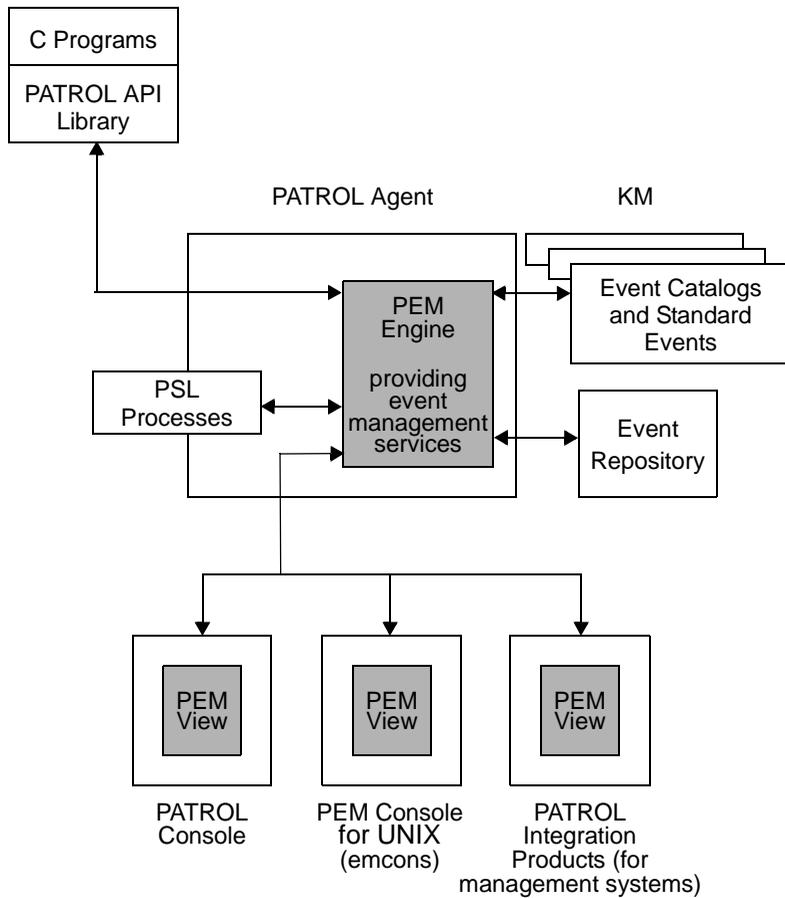
Acknowledging an event can trigger commands (standard or user created). The PEM Engine is configured to determine which events trigger which commands for execution by the PATROL Agent. The agent checks the event catalog to determine whether an Acknowledge command has been defined. If it has, the agent executes the command when the event changes status to acknowledged.

- generate reports of event statistics
- archive events
- send SNMP traps when an event that is configured with `SEND_TRAP` is triggered
- make requests of external SNMP agents and listen for SNMP traps (SNMP management functions)
- respond to SNMP requests from external SNMP managers and send SNMP traps (SNMP agent functions)

# Overview of the PEM Engine

Figure 1-4 on page 1-16 shows how the PEM Engine works with other PATROL components.

**Figure 1-4 The PEM Engine with Other PATROL Components**



# Management Services from the PEM Engine

Use Table 1-2 to determine methods available for requesting event management services from the PEM Engine.

**Note**

Not all PATROL Integration products share the same functionality.

**Table 1-3 Methods to Request Event Management Services**

| To Request this Service...                                 | Use This Method...   |  |  |  |  |  |
|--|--|--|--|--|--|--|
|  | PATROL Console <sup>a</sup>  | PEM Console for Unix <sup>b</sup> ;                                      | PATROL Integration products <sup>c</sup>                                 | PATROL API <sup>d</sup>                                    | PSL <sup>e</sup>   | Command Line   |
| access event knowledge                                     | Event Editor   | N/A  | PemnGetEventClass()<br>PemnGetEventClassList()                           | PemnGetEventClass()<br>PemnGetEventClassList()             | event_catalog_get()  | PatrolCli event get<br>PatrolCli event class                                   |
| archive events and transfer files                          | N/A  | N/A  | N/A  | PemnEventArchive()   | event_archive()<br>remote_file_transfer()                    | PatrolCli event dump   |
| define event knowledge                                     | Event Editor   | N/A  | N/A  | PemnDefineEventClass()<br>PemnSetEventClassCmd()           | N/A  | N/A  |
| display events, event Diary, and event Expert Advice       | Event Details window<br>Event Diary window<br>Event Expert Advice window | Event Details window<br>Event Diary window<br>Event Expert Advice window | Event Details window<br>Event Diary window<br>Event Expert Advice window | PemnAddDiary()<br>PemnReceive()                            | event_query()<br>event_range_query()                         | PatrolCli event listen<br>PatrolCli event query                                |
| filter and query events                                    | Event List window  | Event List window  | Event List window  | PemnReceive()<br>PemnEventRangeQuery()<br>PemnEventQuery() | event_range_query()<br>event_query()<br>remote_event_query() | PatrolCli event query  |
| manage events (for example, with acknowledge/close/delete) | Event List window  | Event List window  | Event List window  | PemnEventManage()<br>PemnEventRangeManage()                | event_range_manage()   | PatrolCli event class<br>PatrolCli event delete<br>PatrolCli event acknowledge |

**Table 1-3 Methods to Request Event Management Services**

| To Request this Service...  | Use This Method...          |                                     |  |                         |   |                        |
|---|-----------------------------|-------------------------------------|--|-------------------------|---|------------------------|
|   | PATROL Console <sup>a</sup> | PEM Console for Unix <sup>b</sup> ; | PATROL Integration products <sup>c</sup> | PATROL API <sup>d</sup> | PSL <sup>e</sup>  | Command Line           |
| obtain a report on Event Statistics   | Event Repository window     | Event Repository window             | Event Repository window                  | PemnGetReport()         | event_report()<br>event_check()                               | PatrolCli event report |
| schedule events   | N/A                         | N/A                                 | N/A                                      | PemnEvent Schedule()    | event_schedule()  | N/A                    |
| sort events   | Event List window           | Event List window                   | N/A                                      | N/A                     | N/A   | N/A                    |
| trigger events specific to your application, events specific to your KM, or user events | N/A                         | Event Trigger dialog                | N/A                                      | PemnSend()              | event_trigger()<br>event_trigger2()<br>remote_event_trigger() | PatrolCli event send   |
|   |                             |                                     |  |                         |   |                        |

<sup>a</sup> Refer to the appropriate *PATROL User Guide*

<sup>b</sup> Refer to the *PATROL Event Manager Console for Unix User Guide*

<sup>c</sup> Refer to the appropriate the *PATROL Integration products documentation set*

<sup>d</sup> Refer to the *PATROL Application Program Interface Reference Manual*

<sup>e</sup> Refer to the *PATROL Script Language Reference Manual*

---

# Starting and Stopping the PATROL Agent

This chapter explains the tasks for properly starting and stopping the PATROL Agent on supported platforms. This chapter contains the following sections:

|   |      |
|---|------|
| Methods for Starting the PATROL Agent . . . . .                                     | 2-3  |
| Default Port Number . . . . .   | 2-3  |
| Startup Process . . . . .   | 2-3  |
| Command Line Arguments for Starting the PATROL Agent . . .                          | 2-4  |
| Methods for Stopping the PATROL Agent . . . . .                                     | 2-8  |
| Stopping the PATROL Process . . . . .   | 2-8  |
| Improper Shutdown . . . . .   | 2-9  |
| PATROL Configuration Utility (pconfig) . . . . .                                    | 2-10 |
| Starting the PATROL Agent on Unix . . . . .   | 2-11 |
| Starting the PATROL Agent for Unix . . . . .  | 2-12 |
| Stopping the PATROL Agent on Unix . . . . .   | 2-13 |
| Stopping the PATROL Agent for Unix with a Script. . . . .                           | 2-14 |
| Stopping the PATROL Agent for Unix with a<br>Configuration Utility Script . . . . . | 2-15 |
| Starting the PATROL Agent on Windows . . . . .                                      | 2-16 |
| PATROL Agent Configuration Utility . . . . .  | 2-16 |
| Modes for Running the PATROL Agent for Windows . . . . .                            | 2-17 |
| Specifying an Account for the PATROL Agent for Windows . . .                        | 2-18 |
| Starting the PATROL Agent for Windows with Services Applet                          | 2-20 |
| Starting the PATROL Agent for Windows from<br>Command Prompt . . . . .              | 2-20 |
| Active Directory Support . . . . .  | 2-22 |

|   |      |
|---|------|
| Publishing to Active Directory . . . . .                                  | 2-22 |
| Information Stored in Active Directory . . . . .                          | 2-23 |
| Stopping the PATROL Agent on Windows . . . . .                            | 2-24 |
| Stopping the PATROL Agent for Windows with<br>Services Applet . . . . .   | 2-24 |
| Stopping the PATROL Agent for Windows from the<br>Command Line . . . . .  | 2-25 |
| Starting the PATROL Agent on OpenVMS . . . . .                            | 2-27 |
| Syntax of PATROL\$STARTUP.COM . . . . .                                   | 2-27 |
| Batch Process Startup of the PATROL Agent on OpenVMS . . . . .            | 2-27 |
| Starting the PATROL Agent on OpenVMS from the<br>Command Line . . . . .   | 2-27 |
| Starting the PATROL Agent on OpenVMS in a Batch Process . . . . .         | 2-29 |
| Stopping the PATROL Agent on OpenVMS . . . . .                            | 2-30 |
| Syntax for PATROL\$SHUTDOWN.COM . . . . .                                 | 2-30 |
| Stop the PATROL Agent on OpenVMS in a Batch Process . . . . .             | 2-30 |
| Stopping the PATROL Agent on OpenVMS from the<br>Command Prompt . . . . . | 2-31 |
| Stopping the PATROL Agent on OpenVMS in a Batch Process . . . . .         | 2-32 |

# Methods for Starting the PATROL Agent

PATROL provides two methods to start the agent. Each method involves a different PATROL interface. You can start a PATROL Agent from

- the command line
- a variety of PATROL components

The platform determines which method you will use to start the PATROL Agent. The agent supports startup scripts for most platforms.

## Default Port Number

Prior to PATROL version 3.4, the default port number was 1987. For version 3.4 and later, it is 3181. If you want to connect to other PATROL products that use the previous default port number, you will have to specify `-p 1987` when starting the agent.

## Startup Process

During startup, the PATROL Agent performs the following processes:

- reads license file
- opens and reads history database
- opens and reads PATROL event logs
- opens PATROL error log
- loads KMs with preloaded statuses
- starts the PATROL SNMP Master Agent (if the variable is set)

You should introduce sleep logic to ensure that all the agent processes are running prior to starting other application and system processes.

# Command Line Arguments for Starting the PATROL Agent

Table 2-1 lists command line arguments for the PATROL Agent.

**Table 2-1 Command Line Arguments for Starting PATROL Agent (Part 1 of 3)**

| Argument   | Function   |
|--|--|
| <code>-config <i>changefile name</i> -batch</code> | <p>processes a pconfig-style configuration file</p> <p>Multiple files are specified by separating them with commas (no intervening spaces). Use standard input if "-" is specified instead of a file name.</p> <p>The -batch argument causes the agent to exit after processing configuration options instead of starting up. It is optional.</p>  |
| <code>-help</code>                                 | displays a list of command line arguments  |
| <code>-i</code>                                    | disables heartbeat   |
| <code>-id <i>name</i></code>                       | <p>The -id option allows you to override the Agent service name that is registered with the RTserver. By default, PATROL Agents register with the RTserver as <i>hostname_portno</i>. For example, starting a PATROL Agent on host nebula with the following command:</p> <pre>PatrolAgent -p 4321 -rtserver tcp:nebula:4321</pre> <p>registers the Agent with the service name NEBULA_4321.</p> <p>Use this feature if you have several hosts with the same name in different domains that are all running PATROL Agents and using the same RTserver. If each host tried to register with the same RTserver using the default service name, only one host could register. The -id option allows you to register an Agent with the RTserver under a different name than the default.</p> |
| <code>-km <i>KM name</i></code>                    | loads initial KM   |

**Table 2-1 Command Line Arguments for Starting PATROL Agent (Part 2 of 3)**

| <b>Argument</b>             | <b>Function</b>  |
|-----------------------------|--|
| -OLMP                       | <p>PSL Optimizer Options: -O[L#][M#][P#]<br/>           Optimization level [L#]<br/> <b>0</b>—no optimization<br/> <b>1</b>—enable level 1 optimization<br/> <b>2</b>—enable level 2 plus level 1 optimization<br/> <b>3</b>—enable level 3 plus level 2 optimization<br/>           Maximum Level Allowed [M#]<br/> <b>0-3</b>—maximum allowable optimization level<br/>           Optimization print level [P#]<br/> <b>0</b>—disable optimization printing<br/> <b>1</b>—print optimization statistics<br/> <b>2</b>— level 1 plus print process instruction execution count<br/> <b>3</b>—level 2 plus print program flowgraph</p> |
| -p <i>portNo</i>            | agent connection port (default port is 3181)   |
| -port <i>portNo</i>         |  |
| -profiling <i>profileNo</i> | <p>activates PSL profiling</p> <p>Valid <i>profileNo</i> are any of the following numbers or a sum of any of those numbers (use the sum to indicate more than one type of profiling):<br/> <b>0</b>—No profiling<br/> <b>1</b>—Process-level profiling<br/> <b>2</b>—Function-level profiling<br/> <b>4</b>—Cumulative<br/> <b>8</b>—Save at exit</p> <p>For more information about PSL profiling, see the <i>PATROL Script Language Reference Manual</i>.</p>   |

**Table 2-1 Command Line Arguments for Starting PATROL Agent (Part 3 of 3)**

| <b>Argument</b>   | <b>Function</b>   |
|---|---|
| -rtserver<br><i>protocol:host.portNo,</i><br><i>protocol2:host2:portNo2</i> | <p>tells the agent which RTserver(s) to connect to, for example: PatrolAgent -rtserver tcp:nebula:2059.</p> <p>The Agent connects to the first RTserver in the list. If the connection breaks, the Agent connects to the next RTserver in the list. You can also use the RTSERVERS environment variable for this purpose.</p> <p>You must define an RTserver if you want to view Agent data from PATROL Central Operator - Microsoft Windows Edition or PATROL Central Operator - Web Edition (PATROL 7.x consoles). If you do not start the Agent with the -rtserver option or set the RTSERVERS environment variable, PATROL 7.x consoles and the PATROL Console Server cannot communicate with the PATROL Agent.</p> |
| -share-sys-output   | shows the same system output window for all consoles that connect to the agent  |
| -v  | <ul style="list-style-type: none"> <li>• reports the OS version number of the machine that the patrolagent.exe is built</li> <li>• shows the version number of the PatrolAgent executable.</li> </ul> <p>accepts upper and lowercase 'v'</p>  |
| -RegServer  | registers COM server and exits  |
| -UnregServer  | unregisters COM server and exits (the PATROL installation automatically registers COM server)   |

## RTSERVERS Environment Variable

The RTSERVERS environment variable tells the agent which RTserver(s) to connect to, for example: tcp:nebula:2059.

The agent connects to the first RTserver in the list. If the connection breaks, the agent connects to the next RTserver in the list.

You must set the RTSERVERS environment variable in either of the following situations:

- You want to view PATROL Agent data from a PATROL 7.x console.
- There is no RTserver installed on a computer in the same subnet.
- The RTserver is not using the default port number of 2059.

The PATROL Console Server, PATROL Agent, and PATROL 7.x consoles can detect RTservers within their own subnets, but they cannot detect RTservers outside of their own subnets. If you have installed one of those components in a subnet that has no RTserver, the only way it can communicate with an RTserver is if you define the RTSERVERS environment variable.

The format of the RTSERVERS environment variable is as follows:

*tcp: computer\_name: port\_number*

where *computer\_name* is the name of a computer running an RTserver (a member of the RTserver cloud), and *port\_number* is the port number that it is using to broadcast.

# Methods for Stopping the PATROL Agent

PATROL provides three methods to stop the agent, and each method uses a different PATROL interface:

- the command line
- a variety of PATROL utilities, particularly the pconfig +KILL option discussed in Chapter 8, “Using pconfig to Configure the PATROL Agent”
- the PATROL Developer Console (click the host icon and select **Developer => Kill Agent**)

## Stopping the PATROL Process

When you stop the PATROL Agent, it performs the following processes:

- writes the process cache
- writes history cache
- waits for child processes to terminate (These may take more time to terminate)
- closes network connections
- stops all other related processes

If you quit the PATROL Agent from an OS shutdown script, you should introduce sleep logic to ensure that the agent completely stops prior to the OS shutdown because the agent waits for child processes to end.

## Improper Shutdown

Quitting the agent improperly can cause problems including corruption of history files and event log files. Following are common methods of improperly stopping the PATROL Agent:

- The computer crashes.
- The computer is shutdown properly, but the PATROL Agent is still running.
- The PATROL Agent process is interrupted using the 'kill -9' command which forces the agent to stop without giving it a chance to close its files.

---

### Note

---

BMC Software recommends that you add logic to stop the PATROL Agent as part of your system shutdown scripts.

---

## Repairing Corrupted History

For more information on repairing corrupt history databases, see “Fixing a Corrupted History Database” on page 12-20.

# PATROL Configuration Utility (pconfig)

The advantages of stopping the agent with the PATROL Configuration utility include:

- It does not require system administrator or root access.
- It provides a standard method that works across all platforms.

---

### Note

---

Access Control List settings can prevent your account from accessing the configuration utility.

---

## Syntax

The basic syntax of the stop command for the pconfig utility is

```
pconfig +KILL [options]
```

## Options

The following table lists the options available for +KILL.

| Option                  | Description  |
|-------------------------|--|
| -host <i>hostname</i>   | stop the agent on a particular host                                |
| -debug                  | print debug information  |
| -lp                     | specify local port in a fire wall that separates utility and agent |
| -port <i>portnumber</i> | stop the agent that is listening on a particular port              |
| +tcp                    | use TCP communication protocol                                     |
| +verbose                | print function calls and results—not as detailed as debug          |

# Starting the PATROL Agent on Unix

The `PatrolAgent` script starts the PATROL Agent on the Unix operating system. You can specify the communication port between the PATROL Console and the PATROL Agent.

The syntax of the `PatrolAgent` command is as follows:

```
./OPT/PATROL/PATROL3/PatrolAgent -p portNo
```

where *portNo* indicates the TCP/IP communications port to use when connecting to the PATROL environment. The port number is optional. If no port number is indicated, use the default port number, 3181.

For a complete list of command line arguments, see “Command Line Arguments for Starting the PATROL Agent” on page 2-4.

## Starting the PATROL Agent for Unix

---

**Summary:** In this task, you will start the PATROL Agent on a Unix machine.

---

### Before You Begin

Before you start the PATROL Agent for Unix, make sure you have performed the following actions:

- Verify that you are logged on the Unix host where you want to start the PATROL Agent.
- Verify that the port you want to use is available. To verify port availability, type the following command in the command line and press **Enter**:

```
netstat -na | grep portNo
```

- Verify that you have changed to the directory containing the PATROL Agent startup script.

### To Start the PATROL Agent for Unix

» Type the following command in the command line and press **Enter**:

```
./PatrolAgent -p portNo
```

A message notifies you that the PATROL Agent is bound to the UDP port and TCP port you specified. The port number allows you to specify a UDP port and a TCP port on which the PATROL Agent will accept connections from PATROL Console. The port number is optional. If you do not specify a port, the PATROL Agent will use the default port number, 3181.

---

**Note**

---

Use the same port for both UDP and TCP.

---

# Stopping the PATROL Agent on Unix

You can create a script to automatically stop the agent during a normal shutdown procedure. However, you do not want to perform an absolute stop (kill -9) because it forces the agent to quit before it can properly close its open files.

---

**Note**

---

The two types of shutdown scripts in the stopping tasks are offered as examples. They are not necessarily robust or complete solutions for every environment.

---

## Stopping the PATROL Agent for Unix with a Script

---

**Summary:** In this task, you will use command line script to stop the PATROL Agent on a Unix host.

---

- » You can use the PATROL Agent process ID to stop the agent. Create a script that resembles the following example:

```
# run this script as root to avoid permission problems
# get the process ID for the PatrolAgent process
pid=`ps -ef | grep PatrolAgent | awk '{print $2}'`
# kill the agent
kill $pid
# allow agent to shutdown before OS removes resources
sleep 2
```

---

### Note

---

The short sleep command gives the agent time to close its files and child processes before the operating system starts to remove resources as part of its normal shutdown procedure. Experiment with your sleep interval. It will vary based on hardware, processes running external to PATROL, number of applications, instances, and parameters running, and other environmental factors.

---

## Stopping the PATROL Agent for Unix with a Configuration Utility Script

---

**Summary:** In this task, you will use a script to call the PATROL configuration utility, `pconfig`, which will stop the PATROL Agent on a Unix host.

---

- » You can use the PATROL utility, `pconfig`, to stop the agent. Create a shell script that resembles the following example:

```
# change to the PATROL installation directory
cd /OPT/PATROL/PATROL3
# run the script to setup your environment. This
# syntax assumes sh or ksh. For csh, use "source
# ../patrolrc"
. ../patrolrc.sh
# change to the binary directory
cd $PATROL_HOME/bin
# execute pconfig to kill the agent
./pconfig +KILL +verbose -port portnumber -host
hostname
# allow agent to shutdown before OS removes resources
sleep 2
```

---

### Note

---

The short sleep command gives the agent time to close up its files and shutdown its child processes before the operating system starts to remove resources as part of its normal shutdown procedure. Experiment with your sleep interval. It will vary based on hardware, processes running external to PATROL, number of applications, instances, and parameters running, and other environmental factors.

---

# Starting the PATROL Agent on Windows

This section describes the following methods of starting the PATROL Agent on Microsoft Windows operating systems:

- the command line
- the Windows Services Applet

The platform determines which method you will use to start the PATROL Agent. The agent supports startup scripts for most platforms.

## PATROL Agent Configuration Utility

You can use the PATROL Agent Configuration utility to configure a PATROL Agent to use a specified user account and select a security library.

If no domain name is specified in the User Name text box, the domain name where the user signed on is used. A domain name can be added explicitly, for example,

*DOMAIN\_NAME\user/encrypted\_password*

The utility checks to see that the specified user account has all the required rights to do the following:

- act as part of the operating system
- increase Quotas
- log on as a service
- replace a process level token
- log on locally
- profile system performance
- debug programs

If the user account does not have the correct rights, a notification dialog is displayed. To modify an account's rights, open the Windows User Manager, modify the rights for the user account accordingly, and run the Configuration utility again.

## Modes for Running the PATROL Agent for Windows

The PATROL Agent for Windows can be run either as a service or as an interactive console application. When run as a service, the agent does not provide textual feedback during its execution. When run as a console application, it is interactive. The agent provides textual feedback during execution, allowing you to perform diagnostics on its operation.

## Specifying an Account for the PATROL Agent for Windows

---

**Summary:** In this task, you will select a user account for the PATROL Agent.

---

### Note

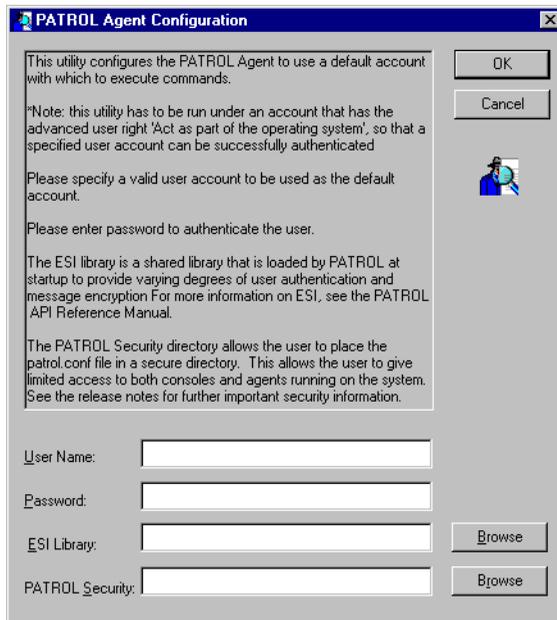
If you have already overridden the /AgentSetup/defaultAccount (creating an agent configuration change file), this task does not have any effect. It changes the value of /AgentSetup/defaultAccount in the config.default file, which is overridden by the value stored in agent configuration change file.

---

**Step 1** Choose **Start => Programs => BMC PATROL => Agent Configure**.

The PATROL Agent Configuration dialog box is displayed.

**Figure 2-1 The PATROL Agent Configuration Dialog Box**



**Step 2** Provide the following information:

- your user name for the host you are running
- your password.

## Starting the PATROL Agent for Windows with Services Applet

---

**Summary:** In this task, you will use the Windows Services utility to start the PATROL Agent on a Windows host.

---

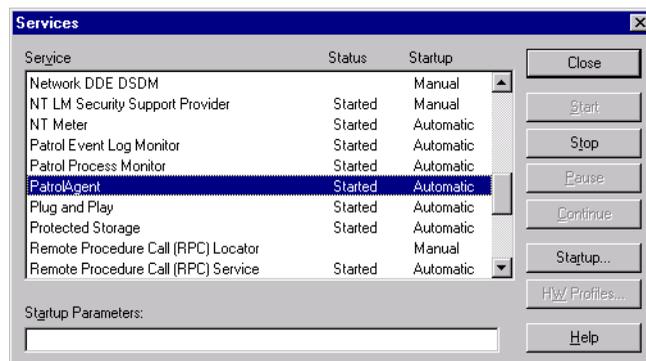
**Step 1** For Windows NT, choose **Start => Settings => Control Panel**.

For Windows 2000, choose **Start => Settings => Control Panel => Administrative Tools**.

**Step 2** Double-click the Services icon.

The Services dialog box is displayed.

**Figure 2-2 Windows NT Services Applet Dialog Box**



**Step 3** For Windows NT, select the PatrolAgent service, enter any command line arguments in the Startup Parameters field, then click Start.

For Windows 2000, select the PatrolAgent service, then select **Action => Properties**, enter any command line arguments in the Start parameters field, then click Start.

The PATROL Agent service starts.

## Starting the PATROL Agent for Windows from Command Prompt

---

**Summary:** In this task, you will start the PATROL Agent on a Windows host.

---

### Before You Begin

Before you start the PATROL Agent for Windows, make sure you have performed the following actions:

- Verify that you are logged on to the host where you want to start the PATROL Agent.
- Verify that no PATROL Agent is already running. From the command line, enter “net start” and press ENTER to view the services that are running.

### To Start the PATROL Agent using the Command Line

» Type the following command in the command line and press **Enter**:

```
PatrolAgent -p portNo
```

A message notifies you that the PATROL Agent is bound to the UDP port and TCP port you specified. The port number allows you to specify a UDP port and a TCP port on which the PATROL Agent will accept connections from PATROL Console. The port number is optional. If you do not specify a port, the PATROL Agent will use the default port number, 3181.

For a complete list of command line arguments, see “Command Line Arguments for Starting the PATROL Agent” on page 2-4.

---

#### **Note**

---

The same port is used for both UDP and TCP.

---

## Active Directory Support

The PATROL Agent supports the Windows 2000 directory service, Active Directory. The Active Directory

- stores information about objects in the enterprise
- uses a structured data store as the basis for a logical, hierarchical organization of directory information
- integrates security through
  - logon authentication
  - access control to objects in the directory

A single network logon provides an authorized network user with access to resources anywhere on the network.

## Publishing to Active Directory

In a homogeneous Windows 2000 environment, each PATROL Agent automatically publishes its existence to the Active Directory during start up. The agent uses the Service Connection Point feature in the Active Directory. When the agent is installed, it creates a serviceConnectionPoint (SCP) object in the Active Directory and sets the attributes. The agent updates these attributes only if either the port number or machine name changes.

PATROL Agent service clients, such as PATROL QuickView, search for a particular agent's SCP object based on SCP keywords. Once the client finds the specified object, it uses the object properties to locate, connect to, and authenticate a PATROL Agent.

# Information Stored in Active Directory

For a PATROL Agent to publish its service to the Active Directory, it must supply

- a port number
- a fully qualified domain name

The agent's information is stored in the following Service Connection Point object's attributes:

**Table 2-2 Service Connection Point attributes set by the PATROL Agent**

| Attribute                 | Description  |
|---------------------------|--|
| keywords                  | the company name and GUID<br><br><b>Example:</b> company name<br>Software/46df95f0-ab3f-4edf-bd80-6be8383c1fed   |
|                           | the PatrolAgent product and GUID<br><br><b>Example:</b><br>PatrolAgent/c5bcd9e0-fb22-45e1-a2f9-990283a96c42)   |
| serviceDNSName            | the DNS name of the service's host computer  |
| serviceDNSNameType        | the serviceDNSName is host name, not SRV recorder  |
| serviceClassName          | the name of patrolagent service<br><br><b>Example:</b> PatrolAgent   |
| serviceBindingInformation | the port number that the agent is listening to<br><br>Once the client locates the SCP object for patrolagent service, it can query for this information and locate the port number, then connect to the agent. |

# Stopping the PATROL Agent on Windows

You can properly stop the PATROL Agent on Windows using one of three methods, each featuring a different utility:

- Windows Services Applet
- command line

## Stopping the PATROL Agent for Windows with Services Applet

---

**Summary:** In this task, you will use the Windows Services utility to stop the PATROL Agent on a Windows host.

---

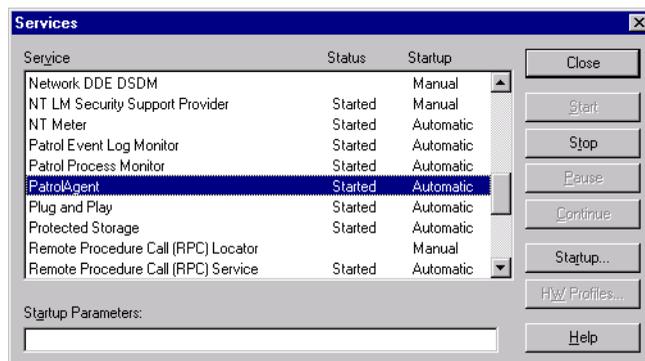
**Step 1** For Windows NT, choose **Start => Settings => Control Panel**.

For Windows 2000, choose **Start => Settings => Control Panel => Administrative Tools**.

**Step 2** Double-click the Services icon.

The Services dialog box is displayed.

**Figure 2-3 Windows NT Services Applet Dialog Box**



**Step 3** For Windows NT, select the PatrolAgent service, then click Stop.

For Windows 2000, select the PatrolAgent service, then select **Action => Stop**.

The PATROL Agent service is stopped. When the computer restarts, the PatrolAgent service starts up again.

## Stopping the PATROL Agent for Windows from the Command Line

---

**Summary:** In this task, you will use the NET STOP command to stop the PATROL Agent on a Windows host.

---

» Type the following command at the command line and press **Enter**:

```
net stop patrolagent
```

The command and its arguments are *not* case sensitive.

A message notifies you that you have stopped the PATROL Agent.

# Starting the PATROL Agent on OpenVMS

The `PATROL$STARTUP.COM` DCL command procedure starts the PATROL Agent on the OpenVMS operating system. You can specify the port at which the PATROL Agent will accept communication from the PATROL Console.

## Syntax of PATROL\$STARTUP.COM

The syntax of the `PATROL$STARTUP.COM` command is as follows:

```
@PATROL$HOME : [MANAGE ] PATROL$STARTUP.COM portNo
```

where *portNo* indicates the TCP/IP communications port to use when connecting to the PATROL environment. If no port number is indicated, the default port number 3181 is used.

If the logical name `PATROL$HOME` has not been defined, use the appropriate path to execute the command; the `PATROL$STARTUP.COM` DCL command procedure then makes all the required logical name definitions.

## Batch Process Startup of the PATROL Agent on OpenVMS

You can start the PATROL Agent for OpenVMS from the command line or in a batch process. For instructions on starting the PATROL Agent for OpenVMS in a batch process, see “Starting the PATROL Agent on OpenVMS in a Batch Process” on page 2-29.

# Starting the PATROL Agent on OpenVMS from the Command Line

---

**Summary:** In this task, you will start the PATROL Agent on an OpenVMS host.

---

## Before You Begin

Before you start the PATROL Agent on OpenVMS, make sure you have performed the following actions:

- Verify that you are logged on to the OpenVMS host where you want to start the PATROL Agent.
- Verify that there is no PATROL Agent already running on the port at which you want to start. To verify, type the following command at the command line and press **Enter**:

```
SHOW SYSTEM/PROCESS=patrolagt*
```

- Verify that you have changed to the directory containing the PATROL Agent startup command file.

## To Start the PATROL Agent on OpenVMS

» Type the following command at the command line and press **Enter**:

```
@PATROL$HOME:[MANAGE]PATROL$STARTUP.COM portNo
```

## Starting the PATROL Agent on OpenVMS in a Batch Process

---

**Summary:** In this task, you will start the PATROL Agent on an OpenVMS host in a batch process.

---

### Before You Begin

Before you start the PATROL Agent on OpenVMS in a batch process, make sure that you start the TCP/IP system first by using:

```
SUBMIT/NOPRINT/USER=PATROL
PATROLDISK4 : [ PATROL . MANAGE ] PATROL$STARTUP /
PARAMETERS=3181
```

If the TCP/IP package is started in a batch process, use the SYNCHRONIZE command to delay the PATROL startup until the TCP/IP package startup has completed.

### To Start the PATROL Agent on OpenVMS

- Step 1** Open the system-specific startup procedure `SYS$MANAGER:SYSTARTUP_VMS.COM` in a text editor.
- Step 2** Insert the following lines into the system-specific startup procedure `SYS$MANAGER:SYSTARTUP_VMS.COM` after the commands used to start the TCP/IP package:

```
SUBMIT/NOPRINT/USER=PATROL
PATROLDISK4 : [ PATROL . MANAGE ] PATROL$STARTUP /
PARAMETERS=3181
```

# Stopping the PATROL Agent on OpenVMS

The `PATROL$SHUTDOWN.COM` DCL command procedure stops the PATROL Agent on the OpenVMS operating system. Because multiple PATROL Agents can run on a single host, you must specify the port on which the agent is running.

## Syntax for PATROL\$SHUTDOWN.COM

The syntax for the `PATROL$SHUTDOWN.COM` command is as follows:

```
@PATROL$HOME : [ MANAGE ] PATROL$SHUTDOWN.COM portNo
```

where *portNo* indicates the TCP/IP communications port used to connect to the PATROL environment.

If the logical name `PATROL$HOME` has not been defined, use the appropriate path to execute the command; the `PATROL$SHUTDOWN.COM` DCL command procedure then makes all the required logical name definitions.

## Stop the PATROL Agent on OpenVMS in a Batch Process

You can stop the PATROL Agent on OpenVMS from the command line or in a batch process. For instructions on stopping the PATROL Agent for OpenVMS in a batch process, see “Stopping the PATROL Agent on OpenVMS in a Batch Process” on page 2-32.

## Stopping the PATROL Agent on OpenVMS from the Command Prompt

---

**Summary:** In this task, you will stop the PATROL Agent on an OpenVMS host.

---

### Before You Begin

Before you stop the PATROL Agent on OpenVMS, make sure you have performed the following actions:

- Verify on which port the PATROL Agent is running. To verify, type the following command at the command line and press **Enter**:

```
SHOW SYSTEM/PROCESS=patrolagt*
```

- Verify that you have changed to the directory containing the PATROL Agent startup command file.

### To Stop the PATROL Agent on OpenVMS

- » Type the following command at the command line and press **Enter**:

```
@PATROL$HOME : [ MANAGE ] PATROL$SHUTDOWN.COM portNo
```

## Stopping the PATROL Agent on OpenVMS in a Batch Process

---

**Summary:** In this task, you will stop the PATROL Agent on an OpenVMS host in a batch process.

---

### To Stop the PATROL Agent for OpenVMS

**Step 1** Open the system-specific shutdown procedure  
SYS\$MANAGER:SYSHUTDOWN\_VMS.COM in a text editor.

**Step 2** Insert the following lines into the system-specific shutdown procedure  
SYS\$MANAGER:SYSHUTDOWN\_VMS.COM

Stop the agent using PATROL logical arguments:

```
@PATROL$HOME:[MANAGE]PATROL$SHUTDOWN 3181
```

Stop the agent using disk and path:

```
@PATROLDISK4:[PATROL.MANAGE]PATROL$SHUTDOWN 3181
```





---

# Background About Configuring the PATROL Agent

This chapter provides background information about configuring the PATROL Agent. It gives an overview of controlling the configuration of the PATROL Agent, the process of configuration, and the methods and options available when configuring. It also provides references to information about common tasks you perform when configuring the PATROL Agent. This chapter discusses the following topics:

|   |      |
|---|------|
| Controlling the Configuration of PATROL Agents . . . . .      | 3-2  |
| Essential Controls for Configuring PATROL Agents . . . . .    | 3-2  |
| Overriding the Default Values . . . . .                       | 3-5  |
| Override Parameters and the PATROL Agent Configuration . . .  | 3-6  |
| Variables that Cannot Be Changed . . . . .                    | 3-7  |
| When Changes to the Configuration Take Effect. . . . .        | 3-7  |
| Process for Configuring the PATROL Agent . . . . .            | 3-9  |
| Methods for Specifying Files and Changing Variables . . . . . | 3-10 |
| Options for Changing Values for Variables . . . . .           | 3-11 |
| Determining the Method for Changing Variables . . . . .       | 3-11 |
| Creating User-Defined Variables . . . . .                     | 3-12 |
| Displaying License File Expiration Warnings . . . . .         | 3-12 |

## Controlling the Configuration of PATROL Agents

PATROL Agent configuration is controlled by the values contained in variables. The variables are contained in configuration files. You can use

- default variables
- customized variables

---

**Note**

---

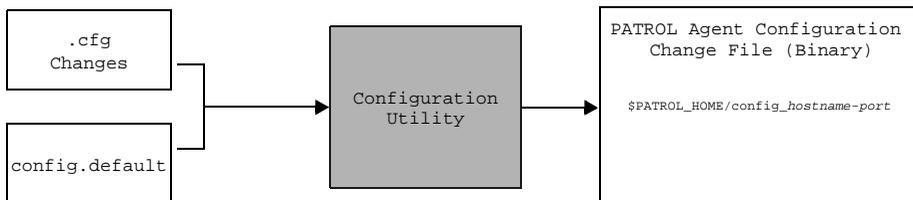
The maximum size of a single agent configuration variable is 1K.

---

## Essential Controls for Configuring PATROL Agents

The configuration utilities combine the contents of the **config.default** and the change file associated with the PATROL Agent. Values for variables in the change file override the values for variables in **config.default**. If a variable is not included in the change file, the PATROL Agent gets its value from **config.default**. Figure 3-1 shows the process flow of configuring agent variables using **config.default** and the change file.

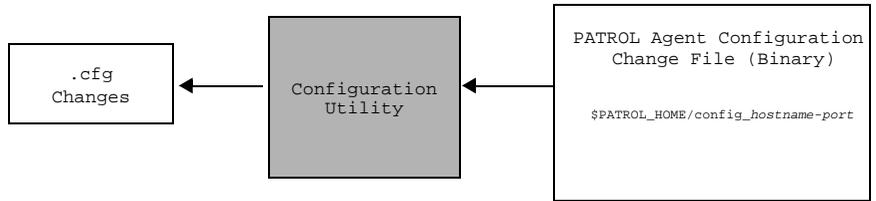
**Figure 3-1** Configuring Agent Variables with the Default and Change File



### Getting Configuration Information from a PATROL Agent

You can use existing configuration information from a PATROL Agent. Using existing information from a PATROL Agent gives you an updated change file. You can now edit or apply any changes. Figure 3-2 shows the process flow to obtain an agent's current configuration information.

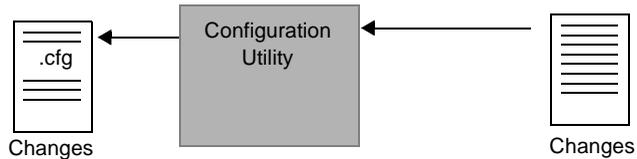
**Figure 3-2 Getting Configuration Information from a PATROL Agent**



### Getting Configuration Information from Another Change File

You can specify the file that has the changes you want to use. You can edit or apply these changes. Figure 3-3 shows the process flow of how to obtain configuration information from a change file.

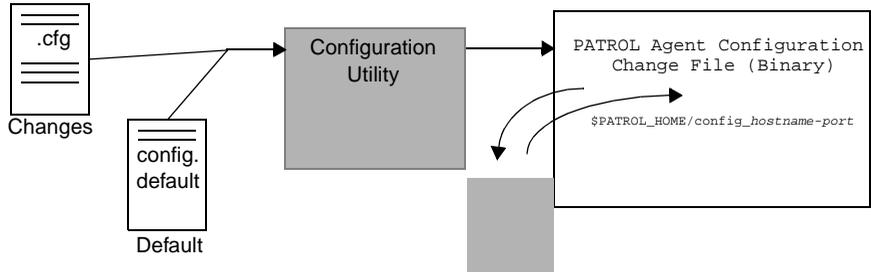
**Figure 3-3 Getting Configuration Information from Another Change File**



## Purging the PATROL Agent Configuration

You can purge the target PATROL Agent configuration files and replace them with new ones constructed from the default configuration, then apply the changes specified in the change file. Figure 3-4 shows the process flow of how to purge an agent's configuration.

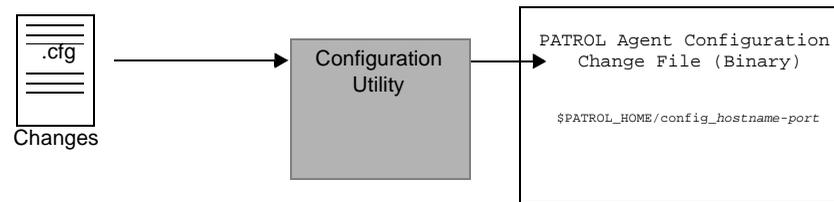
**Figure 3-4 Purging the PATROL Agent Configuration**



## Applying a Change File to an Agent

Figure 3-5 shows the process flow of how you can apply a change file to an agent.

**Figure 3-5 Applying a Change File to an Agent**



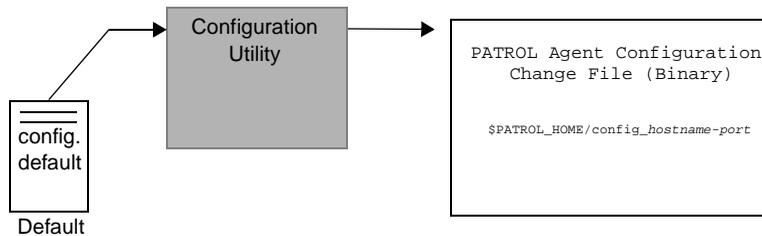
## Reloading the Default Configuration

You can reload the default configuration under the following conditions:

- If the **config.default** file is newer than the existing configuration files, you can reload the default configuration.
- If you want to modify only the underlying default configuration, and you do not want to remove or modify any of the changes that were applied to the configuration, you can reload the default configuration.

The result is a configuration database built from the new default configuration plus the changes that had been previously applied. Figure 3-6 shows the process flow of how you can reload an agent's default configuration.

**Figure 3-6 Reloading an Agent's Default Configuration**



## Overriding the Default Values

You can override the configuration variables in the PATROL Agent. By overriding the variables, you can make sure that multiple PATROL Agents have the same values for all variables, regardless of the version of the **config.default** file that they use.

## Sending Defaults

You can send a complete set of variables to the PATROL Agent to use as the defaults. This causes all variables to be saved in the configuration database as override (non-default) variables. The original defaults for the agent (read from the **config.default** file) are still in the configuration database but are overridden by the defaults you send.

## Getting True Defaults

You can get the complete set of variables to use as the defaults from the PATROL Agent. When you get the true defaults from a PATROL Agent, you get a complete set of variables for the object, including default variables and override (non-default) variables.

## Override Parameters and the PATROL Agent Configuration

In PATROL, override parameters allow an Operator console to make changes to certain attributes of parameters without requiring a Developer console. For instance, changing the alarm ranges of a parameter. These changes supersede properties set by the PATROL KM<sup>™</sup>.

Changes made using override properties are stored in the agent configuration and in the memory value of the parameter. For more information on using override parameters, see Chapter 17, “Managing Parameter Overrides,” or the following books:

- *PATROL Console for Windows 2000 User Guide, II—Monitoring and Managing with PATROL*
- *PATROL Console for Unix User Guide.*

# Variables that Cannot Be Changed

Table 3-1 lists the values for the variables that cannot be changed:

**Table 3-1 Variables That Cannot be Changed**

| Item                     | Variables                      |
|--------------------------|--------------------------------|
| PATROL Agent Setup       | /AgentSetup/_name_             |
| PATROL Agent setup type  | /AgentSetup/_type_             |
| PATROL Agent tuning name | /AgentSetup/AgentTuning/_name_ |
| PATROL Agent tuning type | /AgentSetup/AgentTuning/_type_ |
| SNMP name                | /snmp/_name_                   |
| SNMP type                | /snmp/_type_                   |

## When Changes to the Configuration Take Effect

Every operation that changes the configuration of a PATROL Agent also changes the data and index configuration files as well as the variables in the agent's memory. The new variables take effect immediately. The change does *not* affect variables that must be acted upon at startup.

The following variables do not require the PATROL Agent to be restarted for changes to take effect:

- /AgentSetup/appl.inst.OSdefaultAccount
- /AgentSetup/appl.OSdefaultAccount
- /AgentSetup/appl.OSdefaultAccountAppliesToCmds
- /AgentSetup/application namefilterList
- /AgentSetup/application namefilterType
- /AgentSetup/AgentTuning/agentPriority
- /AgentSetup/AgentTuning/applCheckCycle
- /AgentSetup/AgentTuning/getProcsCycle
- /AgentSetup/AgentTuning/procCachePriority
- /AgentSetup/AgentTuning/procCacheSchedPriority
- /AgentSetup/AgentTuning/psInstructionMax
- /AgentSetup/AgentTuning/psInstructionPeriod
- /AgentSetup/AgentTuning/runqSchedPolicy

- /AgentSetup/AgentTuning/runqDelta
- /AgentSetup/AgentTuning/runqDeltaIncrement
- /AgentSetup/AgentTuning/runqMaxDelta
- /AgentSetup/AgentTuning/userPriority
- /AgentSetup/defaultAccount
- /AgentSetup/disabledKMs
- /AgentSetup/historyRetention
- /AgentSetup/paramHistCacheSize
- /AgentSetup/paramHistCacheTimer
- /AgentSetup/pemPFSnmpEidRange
- /AgentSetup/pemPFSnmpEndTime
- /AgentSetup/pemPFSnmpEvClass
- /AgentSetup/pemPFSnmpNode
- /AgentSetup/pemPFSnmpNseverity
- /AgentSetup/pemPFSnmpOrigin
- /AgentSetup/pemPFSnmpPattern
- /AgentSetup/pemPFSnmpStartTime
- /AgentSetup/pemPFSnmpStatusMask
- /AgentSetup/pemPFSnmpTypeMask
- /AgentSetup/preloadedKMs
- /AgentSetup/preloadedKMsArch
- /snmp/piV1m\_list
- /snmp/sysContact
- /snmp/sysLocation
- /snmp/sysName
- OSdefaultAccount
- OSdefaultAccountAppliesToCmds

---

**Warning**

---

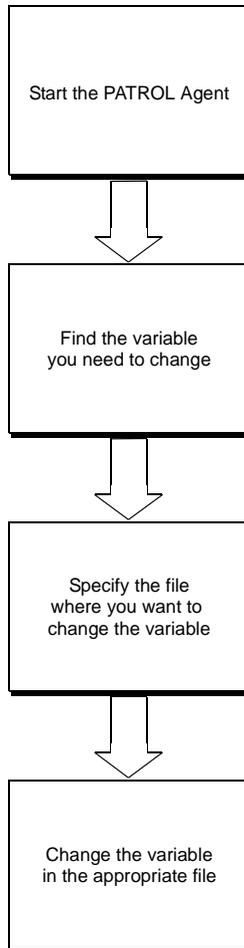
Do not use the PATROL Console to edit the following variables:  
/AgentSetup/defaultAccount and OSdefaultAccount. See “Modifying the  
Default Account Variable” on page 10-18.

---

# Process for Configuring the PATROL Agent

Figure 3-7 shows the process flow for configuring the PATROL Agent by changing the appropriate variables. The variables are contained in configuration files.

**Figure 3-7 Process Flow for Configuring the PATROL Agent**



## Methods for Specifying Files and Changing Variables

Use Table 3-2 to determine the methods available for specifying the files to use and for changing the variables to configure the PATROL Agent.

**Table 3-2 Methods for Specifying Files to Use for Changing Variables**

| <b>Environment</b> | <b>Method for Specifying Files and Changing Variables</b>   |
|--------------------|---|
| UNIX (native)      | <b>pconfig</b> —See Chapter 8, “Using pconfig to Configure the PATROL Agent”  |
| UNIX               | <b>pconfig</b> —See Chapter 8, “Using pconfig to Configure the PATROL Agent.”<br><br><b>xpconfig</b> —See Chapter 9, “Using xpconfig (Unix) to Configure the PATROL Agent,” and Chapter 10, “Using wpconfig (Windows) to Configure the PATROL Agent.” |
| Windows            | <b>wpconfig</b> —See Chapter 10, “Using wpconfig (Windows) to Configure the PATROL Agent.”<br><br><b>pconfig</b> —See Chapter 8, “Using pconfig to Configure the PATROL Agent.”   |

---

### **Warning**

---

Avoid applying changes to more agents than you need when using pconfig, xpconfig, or wpconfig variables.

---

## Options for Changing Values for Variables

Table 3-3 shows the options to use for changing the values for variables.

**Table 3-3 Options for Changing the Values of Agent Variables**

| <b>Task You Want to Perform</b>  | <b>Command to Use in the File</b> |
|--|-----------------------------------|
| Append the specified values with the values for this variable in the agent | APPEND                            |
| Delete the variable text   | DELETE                            |
| Merge the specified values with the values for this variable in the agent  | MERGE                             |
| Replace the values in the agent with the specified values                  | REPLACE                           |

## Determining the Method for Changing Variables

There are two ways you can change the value for a variable:

- manually, using a text editor
- graphically, using the GUI configuration utility

Use Table 3-4 to find instructions on changing variables using the method you prefer.

**Table 3-4 Methods for Changing Variables**

| <b>Method To Change the Variables</b>                                  | <b>Chapter to go to</b>  |
|--|--|
| Manually editing the variable in the change file, using a text editor. |  |
| From a GUI environment, using a GUI configuration utility.             | Chapter 9, "Using xpcnfig (Unix) to Configure the PATROL Agent" or Chapter 10, "Using wpcnfig (Windows) to Configure the PATROL Agent" |

## Creating User-Defined Variables

You can define variables for your own specialized requirements. If you define your own variables, they should be of the form `"/path/variable_name"=value`. If the specific path does not exist, it is created and the variable set equal to *value*.

User-defined variables are stored by the PATROL Agent until their values are requested by Knowledge Module applications, parameters, or SNMP requests. User-defined variables can be completely unrelated to how the PATROL Agent functions. For example, you could specify values for variables such as numerical constants for calculations, lists of names or ID codes, part numbers, and so forth.

## Displaying License File Expiration Warnings

You can use the `/numLicenseDays` variable to control when to display a warning that your license file is about to expire. By default, the following warning is displayed 30 days before the license expires:

“Warning: %d day[s] remaining for PATROL license to expire.”

You can define the warning to be displayed on a day that is less than or equal to the number of days before the license expires as defined in `/numLicenseDays`. You can output the warning to the following locations:

- System Output window
- PATROL Agent error log file
- Unix command prompt

You can get or set this variable by typing the following commands in the system output window:

```
%PSL set("/numLicenseDays", numberofdays);  
%PSL print(get("/numLicenseDays"));
```

---

# Establishing Accounts and Ports

This chapter describes how to set up accounts and ports and specify which applications and instances use which accounts. This chapter discusses the following topics:

|   |     |
|---|-----|
| Accounts . . . . .  | 4-2 |
| Setting the PATROL Agent Default Account . . . . .            | 4-2 |
| Setting the PATROL Agent Default Account Shell . . . . .      | 4-2 |
| Setting the PATROL Agent Account for Applications . . . . .   | 4-3 |
| Adding Time Zones to PATROL . . . . .                         | 4-4 |
| Setting the PATROL Agent Account for Instances . . . . .      | 4-4 |
| Default Accounts for XPC Servers . . . . .                    | 4-5 |
| Using the Application-Specific Account for Commands . . . . . | 4-5 |
| Setting the Default Account for Trusted Clients . . . . .     | 4-7 |
| Ports . . . . .   | 4-8 |
| Setting the Default Port Number on Unix Only . . . . .        | 4-8 |
| Setting a Local Port for Remote Calls . . . . .               | 4-9 |

# Accounts

The PATROL Agent must use a valid account to run parameters, recovery actions, and application discovery. By default, it uses the account used to install PATROL. However, you can change the account that the agent runs under. You can also designate the agent to use particular accounts for applications and instances.

In environments where trusted connections are supported, you can assign an account to be used by a trusted connection that does not have an account on a particular host.

## Setting the PATROL Agent Default Account

The `/AgentSetup/defaultAccount` variable specifies the user account that the agent runs for all parameters, recovery actions, and application discovery procedures when an account is not specified for these commands.

|                                 |                                      |
|---------------------------------|--------------------------------------|
| <b>Formats and Type of Data</b> | text string, not applicable          |
| <b>Default Value</b>            | account used to install PATROL Agent |
| <b>Minimum and Maximum</b>      | not applicable                       |
| <b>Dependencies</b>             | none                                 |
| <b>Recommendation</b>           | none                                 |

## Setting the PATROL Agent Default Account Shell

You can use the `/AgentSetup/defaultAccountShell` configuration variable to specify which shell the PATROL Agent uses for the process spawned by the PATROL Agent default account.

|                                |  |
|--------------------------------|--|
| <b>Format and Type of Data</b> | text string (shell name)   |
| <b>Default Value</b>           | none   |
| <b>Minimum and Maximum</b>     | none   |
| <b>Dependencies</b>            | none   |
| <b>Recommendation</b>          | value must contain a complete path, such as "/bin/sh/" or the PATROL Agent will not work properly. |

This variable is not available in the **config.default** file. You must create this variable manually by using `wconfig` (Windows), `xpconfig` (Unix), or a PSL `pconfig` script such as the following:

```
%PSL pconfig("REPLACE",
"/agentSetup/defaultAccountShell", "new shell");
```

If the variable is set to NULL, the agent defaults to the shell given in the password file of the default account. If the value does not contain a complete path, the PATROL Agent will not work properly.

## Setting the PATROL Agent Account for Applications

The `/AgentSetup/appl.OSdefaultAccount` variable specifies the account that the agent uses when it runs all parameters and recovery actions for this application or application instance. You can override this account by specifying an account in a command.

|                                 |   |
|---------------------------------|---|
| <b>Formats and Type of Data</b> | text string, not applicable             |
| <b>Default Value</b>            | NULL (use PatrolAgent's defaultAccount) |
| <b>Minimum and Maximum</b>      | not applicable                          |
| <b>Dependencies</b>             | none                                    |
| <b>Recommendation</b>           | none                                    |

## Adding Time Zones to PATROL

You can use the `/AgentSetup/timeZone` configuration variable to define time zones that do not exist in the PATROL Agent time zone table. If you are using a system that runs in a time zone not recognized by PATROL, you can add the time zone and its offset to the `/AgentSetup/timeZone` variable. When the PATROL Agent finds the `timeZone` variable, it will add the contents to its time zone table and use the new time zone(s) data to calculate date and time.

|                                |   |
|--------------------------------|---|
| <b>Format and Type of Data</b> | TimeZone/OffsetValue<br>TimeZone= the name of the time zone<br>OffsetValue = the offset value, in minutes, for the specified time zone. Begin the OffsetValue with a + or - sign. |
| <b>Default Value</b>           | NULL  |
| <b>Minimum and Maximum</b>     | none  |
| <b>Dependencies</b>            | none  |
| <b>Recommendation</b>          | none  |

If you define the `timeZone` variable as `TZ1/+200,TZ2/-100`, the PATROL Agent recognizes `TZ1` and `TZ2` as time zone strings and will use their corresponding offset values for date conversion calculation. By default, the value of the variable is `NULL`.

## Setting the PATROL Agent Account for Instances

The `/AgentSetup/appl.inst.OSdefaultAccount` variable specifies the account that the agent uses when it runs all parameters and recovery actions for this application instance. You can override this account by specifying an account in a command.

|                                 |   |
|---------------------------------|---|
| <b>Formats and Type of Data</b> | text string, not applicable             |
| <b>Default Value</b>            | NULL (use PatrolAgent's defaultAccount) |

|                            |                |
|----------------------------|----------------|
| <b>Minimum and Maximum</b> | not applicable |
| <b>Dependencies</b>        | none           |
| <b>Recommendation</b>      | none           |

## Default Accounts for XPC Servers

You can use the `/AgentSetup/XPC/<xpcserver>.xpc_defaultAccount` to specify a default account for each xpc server. When the xpc process is spawned in PatrolAgent, it checks to see if there is a default account for that xpc process and switches the user to that account before executing that process. If the account is not defined, the xpc server runs under the PATROL Agent default account.

|                                |   |
|--------------------------------|---|
| <b>Format and Type of Data</b> | text string<br><xpcserver>.xpc_defaultAccount |
| <b>Default Value</b>           | none  |
| <b>Minimum and Maximum</b>     | none  |
| <b>Dependencies</b>            | none  |
| <b>Recommendation</b>          | none  |

---

### Note

---

This variable is available only for Microsoft Windows platforms.

---

## Using the Application-Specific Account for Commands

The `/AgentSetup/appl.OSdefaultAccountAppliesToCmds` variable determines whether menu commands run against instances of this application use the account specified by either `appl.inst.OSdefaultAccount` or `appl.OSdefaultAccount`. Otherwise, menu commands use the account with which the console logs into the agent. You can override this account by specifying an account in a command.

|                                 |                                       |
|---------------------------------|---------------------------------------|
| <b>Formats and Type of Data</b> | boolean, yes or no                    |
| <b>Default Value</b>            | no (do not run as *.OSdefaultAccount) |
| <b>Minimum and Maximum</b>      | not applicable                        |
| <b>Dependencies</b>             | none                                  |
| <b>Recommendation</b>           | none                                  |

## Setting the Default Account for Trusted Clients

The `/AgentSetup/trustedConnectionsAccount` variable specifies the default account that the agent assigns to a trusted client connection that does not have an account on the box. First, the agent tries the account for the trusted user. If the account is not available, the agent uses the account specified by `trustedConnectionsAccount`.

---

### Note

---

This variable is applicable only to installations using supported external authentication services, currently Kerberos 5.

---

|                                 |   |
|---------------------------------|---|
| <b>Formats and Type of Data</b> | text string, not applicable   |
| <b>Default Value</b>            | patrol<br><br>If the agent fails to get the client's account, and this variable is not set or set to a non-valid account, the agent rolls back to "username/encryptedPassword" scheme used on non-trusted connections |
| <b>Minimum and Maximum</b>      | not applicable  |
| <b>Dependencies</b>             | none  |
| <b>Recommendation</b>           | none  |

# Ports

The agent allows you to specify the default port on Unix and specify the port used to communicate through a firewall in a secure environment.

PATROL also allows you to establish ports for sending and receiving SNMP trap information.

## Setting the Default Port Number on Unix Only

The **PATROL\_PORT** environment variable specifies the PATROL Agent port number on Unix only. This variable applies only if the port number has not been specified by another means.

### Hierarchy for Unix Agent Port Number

Following is the hierarchy for the PATROL Agent port number on Unix:

- 1.) -p command line option is specified, use command line value
- 2.) "patrolagentsubagent" defined in /etc/services
- 3.) environment variable PATROL\_PORT
- 4.) hard-coded default (3181)

## Setting a Local Port for Remote Calls

The `/AgentSetup/localPortForRemoteOpen` variable specifies the local UDP port number for the agent. The PSL `remote_open()` function uses this information to work through a firewall.

|                                 |                      |
|---------------------------------|----------------------|
| <b>Formats and Type of Data</b> | numeric, port number |
| <b>Default Value</b>            | " " (not defined)    |
| <b>Minimum and Maximum</b>      | not applicable       |
| <b>Dependencies</b>             | none                 |
| <b>Recommendation</b>           | none                 |

# Managing Console Connections

This chapter describes how to manage the PATROL Agent's relationship with various consoles and utilities, and control access to the agent. This chapter discusses the following topics:

|   |      |
|---|------|
| The Agent-Console Relationship . . . . .  | 5-2  |
| Displaying and Logging PEM Connection Messages . . . . .                                      | 5-2  |
| Changing the Unix Display Environment . . . . .   | 5-3  |
| Controlling System Output Window Display . . . . .  | 5-3  |
| Setting the Agent to Run without a Console Connection . . . . .                               | 5-4  |
| Designating an IP Address for Connections on Systems with<br>Multiple Network Cards . . . . . | 5-5  |
| Recognize Additional IP Addresses. . . . .  | 5-7  |
| Connections Using DHCP . . . . .  | 5-8  |
| Controlling Access to the Agent. . . . .  | 5-9  |
| Backing Up Your Configuration Files Before Defining an ACL. . . . .                           | 5-9  |
| Defining Access Control Lists . . . . .   | 5-9  |
| Enabling System Output Window Display . . . . .   | 5-13 |
| HostName and UserName Attribute Conventions . . . . .   | 5-13 |
| Connection Modes and Accounts . . . . .   | 5-15 |
| Required Access. . . . .  | 5-16 |
| Restoring Access in Case of Lockout . . . . .   | 5-16 |
| Examples of ACL Usage . . . . .   | 5-17 |
| Access for Agent Configuration Utility Started through the<br>Developer Console . . . . .     | 5-18 |
| Access for Agent Configuration Utility Started from the<br>Command Line . . . . .             | 5-19 |
| Access for PATROL Event Manager Console . . . . .   | 5-19 |
| Access for the Knowledge Module Scripts . . . . .   | 5-20 |

|                                       |      |
|---------------------------------------|------|
| Access for PEM Applications . . . . . | 5-21 |
| Error Messages . . . . .              | 5-21 |

# The Agent-Console Relationship

The PATROL Agent can manage several aspects of its relationship with various types of PATROL Consoles and other utilities. Some aspects include:

- what types of consoles connect to it
- what types of PEM messages the agent sends to the PEM Console
- where Unix applications display their information
- how the agent behaves if no consoles connect to it
- who can connect to it

## Displaying and Logging PEM Connection Messages

The `/AgentSetup/suppressConsoleInfoMsgMask` variable controls whether PEM messages are logged in the PEM Log and are displayed in the PEM Console or excluded from the log and not displayed in the PEM Console. Suppressing messages is sometimes necessary because various applications generate numerous events.

|                            |  |
|----------------------------|--|
| <b>Values</b>              | Specify the connection modes for which the Agent blocks informational messages.<br><b>C</b> —configuration<br><b>D</b> —developer<br><b>O</b> —operator<br><b>P</b> —PEM |
| <b>Default Value</b>       | P (PEM)  |
| <b>Minimum and Maximum</b> | not applicable   |
| <b>Dependencies</b>        | none   |
| <b>Recommendation</b>      | To view all messages and record them in the log file, set this variable equal to an empty string ("").   |

## Changing the Unix Display Environment

The */AgentSetup/defaultDisplay* variable specifies where Unix applications display their output. If a parameter, recovery action, or application discovery procedure runs an X windows application and the agent is not associated with a terminal, the application uses this display value to determine where to send its output.

|                                |   |
|--------------------------------|---|
| <b>Format and Type of Data</b> | text string (no spaces), not applicable                         |
| <b>Default Value</b>           | :0 (colon zero, which is not defined)                           |
| <b>Minimum and Maximum</b>     | not applicable  |
| <b>Override</b>                | DISPLAY environment variable                                    |
| <b>Dependencies</b>            | none  |
| <b>Recommendation</b>          | In most circumstances, you do not need to change this variable. |

## Controlling System Output Window Display

You can use the */AgentSetup/suppressSystemOutputMsgMask* configuration variable to suppress or allow the display of KM-related messages in the system output window.

|                                |                   |
|--------------------------------|-------------------|
| <b>Format and Type of Data</b> | Boolean (yes, no) |
| <b>Default Value</b>           | no                |
| <b>Minimum and Maximum</b>     | none              |
| <b>Dependencies</b>            | none              |
| <b>Recommendation</b>          | none              |

## Setting the Agent to Run without a Console Connection

The */AgentSetup/sessionsInitMayFail* variable determines whether the PATROL Agent will run if no sessions to the PATROL Console can be established.

|                                |  |
|--------------------------------|--|
| <b>Format and Type of Data</b> | Boolean, yes or no   |
| <b>Default Value</b>           | no   |
| <b>Minimum and Maximum</b>     | not applicable   |
| <b>Dependencies</b>            | none   |
| <b>Recommendation</b>          | <p>In most circumstances, you do not need to change this variable.</p> <p>Set this variable to yes on remote agents. This enables the agent to monitor an application and collect information from the applications' inception. It ensures that the history is complete. The console does not have to connect to the remote agent until it is ready to retrieve the agent's history.</p> |

---

### Note

---

This variable does not apply for PATROL Central Operator – Microsoft Windows Edition or PATROL Central Operator – Web Edition.

---

# Designating an IP Address for Connections on Systems with Multiple Network Cards

If an agent is running on a machine that has multiple network interface cards (perhaps one dedicated to a private network and another dedicated to a public network), you must instruct the agent about which one to use when connecting to consoles and other utilities.

---

## Note

---

You must create the **BindToAddress** and **PortConnectType** variables. They are not created during the installation of the PATROL Agent. See Appendix A for more information about these variables.

---

## Specify Network Card IP Address

The **/AgentSetup/BindToAddress** configuration variable specifies which IP Address, and thus which network card, the PATROL Agent uses to communicate with consoles or other utilities in a system with multiple network cards.

|                                |   |
|--------------------------------|---|
| <b>Format and Type of Data</b> | numeric, IP address   |
| <b>Default Value</b>           | none<br><br>The PATROL Agent uses any IP address available. If more than one IP address is available, PATROL uses both. This results in intermittent connections. |
| <b>Minimum and Maximum</b>     | not applicable  |
| <b>Dependencies</b>            | AgentSetup/PortConnectType determines the communication protocol used when establishing a connection through the port specified by this variable                  |
| <b>Recommendation</b>          | If you type in the wrong IP address (make a typographical error), the agent does not start.   |

## Incorrect IP Address

If the IP address is incorrect, the agent does not start. Instead, it writes error messages, similar to the following ones, to the PATROL Agent Error Log.

```
DAY MON DD HH:MM:SS CCYY PatrolAgent-F-EINTERNAL: Sess Init failed:
err = -14/10049 Another agent is probably running on the same port
ID 102061: DAY MON DD HH:MM:SS CCYY: Exiting due to error of type
EINTERNAL
ID 102062: DAY MON DD HH:MM:SS CCYY: PatrolAgent terminating
```

## Set Communication Protocol

The **/AgentSetup/PortConnectType** configuration variable specifies the communication protocol for a connection established through the port specified by the **BindToAddress** configuration variable.

|                            |  |
|----------------------------|--|
| <b>Value</b>               | The valid communication protocols for establishing a connection include:<br><b>TCP</b> —use only TCP<br><b>UDP</b> —use only UDP<br><b>BOTH</b> —use either TCP or UDP |
| <b>Default Value</b>       | BOTH   |
| <b>Minimum and Maximum</b> | not applicable   |
| <b>Dependencies</b>        | AgentSetup/BindToAddress sets the IP address for which PortConnectType determines the communication protocol   |
| <b>Recommendation</b>      | none   |

## Recognize Additional IP Addresses

The `/AgentSetup/IPAddresses` configuration variable lists all the possible IP addresses through which the agent can communicate. This variable enables the agent to run on and communicate from a server with multiple network interface cards (NIC). The agent will not accept packets that originate from IP Addresses that it does not recognize.

|                                |                     |
|--------------------------------|---------------------|
| <b>Format and Type of Data</b> | numeric, IP address |
| <b>Default Value</b>           | none                |
| <b>Minimum and Maximum</b>     | not applicable      |
| <b>Dependencies</b>            | none                |
| <b>Recommendation</b>          | none                |

# Connections Using DHCP

PATROL components can connect and operate in a Windows environment that uses Dynamic Host Configuration Protocol (DHCP). However, the PATROL components and the DHCP environment must be properly configured.

## PATROL Console Conditions

For the PATROL Console to run in a DHCP environment, the console must reference the host where the PATROL Agent is running by host name. If an agent shuts down and the DHCP server reassigns its IP address, the console can reconnect because the host name remains the same.

---

### Warning

---

The host where the PATROL Console is running must support “nslookup” and “ping” features by both host name and IP address.

---

## PATROL Agent Conditions

For the PATROL Agent to run in DHCP environment, the environment must meet the following conditions:

- The PATROL Agent must be able to obtain its own IP address during startup.
- The IP address of the PATROL Agent cannot change while the agent is running.

The agent stores and uses its IP address to execute internal and external commands. Another reason for these conditions is that the session layer rejects all packets with IP addresses that the agent does not recognize.

## Reconnecting

If an agent shuts down and the DHCP lease expires, the agent may lose its current IP address and be reassigned another one by the DHCP server. To reconnect, manually restart the agent and update the connection from the PATROL Console (use MB3 to choose Update Connection).

# Controlling Access to the Agent

Through an access control list (ACL), the agent allows you to define the following:

- who has access to the agent
- what hosts have access to the agent
- what type of PATROL consoles and utilities have access to the agent and any combination of these three types of control.

---

## Note

---

The actual log ons are subject to account and password validation. Even with the most lenient access control list (all users from all hosts in any type of mode), you cannot log on with an unknown account.

---

## Backing Up Your Configuration Files Before Defining an ACL

Before creating an ACL, back up your Agent Configuration File (**config.default**), which contains installation default values, and your Agent Configuration Change File (*user-defined\_filename.cfg*), which contains any customizations that you've made to agent configuration variables. Copy and save these files outside the PATROL directory structure.

Backing up these files ensures that you can restore the agent to its previous state if your ACL definitions produce unexpected or undesirable results.

## Defining Access Control Lists

The **/AgentSetup/accessControlList** variable controls which users are authorized to connect to an agent in which modes from which hosts.

|                                       |  |
|---------------------------------------|--|
| <p><b>Format and Type of Data</b></p> | <p>For each access control list (ACL), the format is a comma-separated list of entries. Each entry has the following format:</p> <p><i>UserName/HostName/Mode</i></p> <p><b>UserName</b>—the name of a local account that the connecting console may request to use. It defaults to *. For more information, see “HostName and UserName Attribute Conventions” on page 5-13.</p> <p><b>HostName</b>—a machine (console) that is authorized to connect to this agent. It defaults to *. For more information, see “HostName and UserName Attribute Conventions” on page 5-13.</p> <p><b>Mode</b>—a list of application and application modes that are authorized to access the agent.</p> <ul style="list-style-type: none"> <li><b>C</b>—Configure (pconfig, wpconfig, xpcconfig)</li> <li><b>D</b>—Developer (console)</li> <li><b>O</b>—Operator (console)</li> <li><b>P</b>—PEM (event manager console)</li> <li><b>R</b>—Allow operator overrides</li> <li><b>S</b>—System Output Window Display</li> </ul> <p>See Table 5-1, on page 5-11</p> <p><b>Note</b><br/>If the Mode value is missing from an individual ACL entry, it defaults to O.</p> |
| <p><b>Default Value</b></p>           | <p>*/*/CDOPS</p>   |
| <p><b>Minimum and Maximum</b></p>     | <p>not applicable</p>  |
| <p><b>Dependencies</b></p>            | <p>none</p>  |
| <p><b>Recommendation</b></p>          | <p>See the following sections</p>  |

The following table explains the agent connection modes:

**Table 5-1 Access Control List Connection Modes**

| Mode    | Description  |
|---------|--|
| C       | <p>The C (configure) mode controls the context of commands that are executed on the PATROL Agent. The following commands are executed using the console connection account:</p> <ul style="list-style-type: none"> <li>• commands executed from the System Output window</li> <li>• KM menu commands</li> <li>• agent configuration commands</li> <li>• OS commands and tasks</li> <li>• PSL commands and tasks</li> </ul> <p><b>Note</b><br/>If a command is executed from the system command line, the command executes using the credentials of the user logged-on to the system.</p> |
| D and O | <p>The D (developer) and O (operator) connection modes control the connection type between the agent and console:</p> <ul style="list-style-type: none"> <li>• D allows a developer connection to the agent</li> <li>• O allows an operator connection to the agent</li> </ul> <p><b>Note</b><br/>The user must have rights to log on locally to the agent system to connect with the console.</p>   |
| P       | <p>The P (PEM) connection mode controls access to the agent using the PEMAPI. The P mode does not control the availability of the PATROL Event Manager. The P mode controls the following types of access to the PATROL Agent:</p> <ul style="list-style-type: none"> <li>• access from applications that use PEMAPI functions to connect to the PATROL Agent</li> <li>• connection to the PATROL Agent using the remote PSL functions</li> </ul>  |

**Table 5-1 Access Control List Connection Modes**

| <b>Mode</b> | <b>Description</b>  |
|-------------|---|
| R           | The R mode allows operator overrides on agents and consoles only if the following variable is set to <b>true</b> in <b>patrol.conf</b> :<br>allowoverrideparameter<br><br><b>Note</b><br>This mode is only available for 3.5 agents and consoles. |
| S           | The S (System Output Window Display) mode allows display of system output window if the following variable is set to 1:<br>/AgentSetup/EnableSysOutputAclCheck  |

---

**Note**

---

The defaultaccount should have C mode access to the PATROL agent.

---

## Changing Connection Mode Behavior

By default, the PATROL Agent runs discovery, collection, and recovery actions as the defaultaccount, and commands executed from the System Output window, InfoBoxes, and menus are executed using the console connection account.

The default behavior is changed by using the following PATROL Agent variables:

```
/AgentSetup/appl.OSdefaultAccount" = {  
  REPLACE="user" },
```

```
/AgentSetup/appl.inst.OSdefaultAccount" = {  
  REPLACE="user" },
```

```
/AgentSetup/appl.OSdefaultAccountAppliesToCmds" = {  
  REPLACE="no" },
```

## Enabling System Output Window Display

You can use the */AgentSetup/EnableSysOutputAclCheck* configuration variable to control whether S mode of the */AgentSetup/accessControlList* variable can allow a user to view the system output window. If *EnableSysOutputAclCheck* is set to 1 and a user account has S mode permissions set in *accessControlList*, then the user is allowed to display the System Output window.

|                                |               |
|--------------------------------|---------------|
| <b>Format and Type of Data</b> | Boolean (1,0) |
| <b>Default Value</b>           | 1             |
| <b>Minimum and Maximum</b>     | none          |
| <b>Dependencies</b>            | none          |
| <b>Recommendation</b>          | none          |

## HostName and UserName Attribute Conventions

In an ACL entry, you can use a number of masking techniques for the host name and user name attributes.

---

### Note

Use the same naming convention in the ACL host name that you used when setting up connections to agents. If you used short names in the connections, use short names in the ACL entries. If you used fully qualified names in the connections, use fully qualified names in the ACL entries. The ability of PATROL to associate a short name with a fully qualified name depends upon how your DNS server is set up. Using the same naming convention ensures that your ACL entries work properly regardless of how your DNS server is set up.

---

### UserName

The name of a local account that the connecting console may request to use. Valid values include

\*—any username (assuming the account exists)

**username**—an actual name of an account

If the HostName value is not provided for an ACL entry, it defaults to \*.

## HostName

A machine (console) that is authorized to connect to this agent. You can specify a hostname by using the fully qualified name, the short name, or a partial name (pattern) created with a wildcard specification in which the first character is a '\*', with other characters following.

\*—any host name (assuming the host exists)

**hostname**—an IP Address, range of addresses, or actual name of the host indicating that this entry is for that host only

To define a range of IP Addresses for the **hostname** value, define any string in the form of and IP Address and specify what bits need to be allowed, such as A.B.C.D|e in which e is a number between 0 and 32 that specifies which bits are set in a 32-bit number.

Examples:

\*/172.19.0.0|16/CDOP

All hosts with IP Addresses matching the first 16 bits (172.19) are connected.

\*/172.19.20.30|24/CDOP

All hosts with IP Addresses from 172.19.20.0 – 172.19.20.255 are connected.

\*/172|8/CDOP

All hosts whose IP Addresses start with 172 are connected.

\*/172.19.|16/CDOP

All hosts whose IP Addresses start with 172.19. are connected.

\***parital\_hostname**—a wildcard specification, in which the first character is an asterisk followed by other characters.

If the HostName value is not provided for an ACL entry, it defaults to \*.

## Connection Modes and Accounts

Table 5-2 describes how the various consoles and utilities connect to the agent and what type of account each uses. The accounts to connect to the agent include

- connection account—account used to connect 3.x consoles to the PATROL Agent and 7.x consoles to the PATROL Console Server
- default account—account stored in /AgentSetup/defaultAccount agent configuration variable
- system log-on account—account used to log on to the operating system and used to access the PATROL Console

**Table 5-2 Connection Modes and Accounts Used by the Agent's Clients**

| Client   | Mode | Account Used   |
|--|------|--|
| developer console  | D    | connection account   |
| <ul style="list-style-type: none"> <li>• operator console</li> <li>• PATROL Central Operator - Microsoft Windows Edition</li> <li>• PATROL Central Operator - Web Edition</li> </ul> | O    |  |
| <ul style="list-style-type: none"> <li>• xpconfig</li> <li>• wpconfig</li> <li>• pconfig</li> </ul>  | C    | <p>When started from the command line, these utilities use the system log-on account.</p> <p>When started from within a developer console, these utilities use the system log-on account.</p>                                      |
| pconfig( )   | C    | <p>When this function is run by a parameter, recovery action, or application discovery, it uses the default account.</p> <p>When this function is run by a Menu command or an Infobox command, it uses the connection account.</p> |
| PATROL Event Manager (PEM)   | P    | system log-on account  |
| User-coded client that uses PATROL API   |      |  |

## Required Access

Be careful not to restrict the agent, which uses the default account (/AgentSetup/defaultAccount), or the configuration utilities (xpconfig, wpconfig, and pconfig) from performing configuration changes on the agent.

To be able to configure an agent and run knowledge module scripts that contain the pconfig() function, the default account must have configuration access from the host on which the agent is running.

```
"/AgentSetup/accessControlList" = {  
REPLACE="defaultAccount/*/C,logonAccount/*/C " }
```

## Restoring Access in Case of Lockout

If the ACL is improperly set up, all user accounts can be denied access to the agent.

To restore access to the agent, you must remove the Agent Configuration Change File from the PATROL directory structure. For information about the Agent Configuration Change File, see “Essential Controls for Configuring PATROL Agents” on page 3-2.

---

### Warning

---

If you delete the Agent Configuration Change File rather than move it, you will lose all customizations made to the file.

---

## Examples of ACL Usage

These examples illustrate common uses of access control lists.

### Allow a User From Any Machine that Matches the Pattern

To permit any host whose name ends in “.acme.com” to connect as an operator console and log on as user “safeuser”, you would enter

```
"/AgentSetup/accessControlList" = {  
  REPLACE="safeuser/*.acme.com/O" }
```

### Allow Any User From a Machine

To permit any user on the machine “secure.acme.com” to connect in either developer or operator mode and log on with any existing account, enter

```
"/AgentSetup/accessControlList" = {  
  REPLACE="*/secure.acme.com/DO" }
```

### Allow Any Administrator on the Administrator Machine or an Administrator on Any Machine that Matches the Pattern

To permit any administrator (whose name begins with “Admin”) on the machine “admin.acme.com” to connect in developer, operator, configuration mode or to permit an administrator (Admin\_3) to logon from any machine in the \*.acme.com domain in configuration mode, enter

```
"/AgentSetup/accessControlList" = { REPLACE  
  ="Admin*/admin.acme.com/DOC,Admin_3/*.acme.com/C" }
```

## Access for Agent Configuration Utility Started through the Developer Console

When you access the PATROL Agent with the Agent Configuration utility through the PATROL Developer Console (use MB3 to choose Development => Agent Configuration), the utility uses the account that was used to log on to the operating system. The utility does not use the account that the console used to connect to the agent.

---

### Warning

---

At least one account, the account that is used to log on to the console, must have developer and configuration access mode.

---

To perform development and configuration actions, the following two conditions must be met:

- The account's ACL entry must include both Developer ("D") and Configure ("C") access.
- The user must log on to the operating system and the developer console using the same account.

To permit a user (patrol\_adm\_4) to perform configuration and development tasks from any machine, enter

```
"/AgentSetup/accessControlList" = { REPLACE =  
"patrol_adm_4/*/*CD" }
```

## Access for Agent Configuration Utility Started from the Command Line

When either pconfig, wpconfig, or xpconfig is started from the command line, the operating system logon account is used to identify the session. Similarly, when xpconfig or wpconfig is started from a console or the Agent's system output window on a console, the same rule applies.

---

### Note

---

You cannot start the wpconfig utility from the system output window on PATROL Console for Windows. To start wpconfig, use MB3 to choose Development => Agent Configuration.

---

## Access for PATROL Event Manager Console

The PATROL Event Manager Console uses different accounts depending upon how it is started.

### Started from Console

#### Unix

The PATROL Event Manager Console is accessed through the PATROL Console and it shares the console's connection to the agent. If the console can connect to the agent, the ACL cannot prevent the Event Manager Console from receiving events.

#### Windows

The PATROL Event Manager Console is built into the PATROL Developer Console for Windows. The PEM Console does not establish a separate connection with the agent and is not affected by ACL entries that contain the PEM access mode (P).

When the PATROL Event Manager Console uses the PATROL Console's connection to the agent, it does not register as a PEM application and is therefore not affected by the PEM mode in an ACL entry.

## Started from Command Line

### Unix

When the Patrol Event Manager Console is started from the command line as a separate process, independent of the console, the PEM Console uses the OS logon account. When started from the command line, you can restrict its access to the agent by excluding the PEM mode value from ACL entry.

### Windows

The PATROL Event Manager Console application cannot be started from the command line. The application is built into the PATROL Console for Windows.

## Access for the Knowledge Module Scripts

Many application Knowledge Modules use `pconfig()` in their PSL scripts to edit and add agent configuration variables. The PSL `pconfig()` function uses the PATROL Agent default account (`/AgentSetup/defaultAccount`).

To ensure that PSL `pconfig()` function executes successfully, assign the default account Configuration access, (C).

## Access for PEM Applications

If the access control list does not include PEM access (P), the agent will deny access to all applications that communicate with the PATROL Agent through the PEM API including:

- PATROL<sup>®</sup> Adapter for Microsoft Office
- PATROL<sup>®</sup> QuickView
- PATROL<sup>®</sup> Integration products

## Error Messages

When the agent denies a connection based upon the access control list, it writes an application-specific error message.

### PATROL Console

When based upon the access control list, the agent denies a connection attempt from the PATROL Console (operator or developer), the agent sends the following message to the console's System Output window:

```
Not authorized to connect to this agent.  
Connection with 'hostname' on port portNo was shutdown  
by the agent.
```

## Controlling Events Displayed in PEM

You can use the *allowsendparamonly* variable to prevent events from being generated and displayed in the PATROL Event Manager for applications and instances when parameters change state to OK. The *allowsendparamonly* is defined in the [AGENT] stanza of the **patrol.conf** file.

|                                |                           |
|--------------------------------|---------------------------|
| <b>Format and Type of Data</b> | text string (true, false) |
| <b>Default Value</b>           | none                      |

|                            |      |
|----------------------------|------|
| <b>Minimum and Maximum</b> | none |
| <b>Dependencies</b>        | none |
| <b>Recommendation</b>      | none |

The **allowsendparamonly** variable has the following values:

- true: events are not generated or displayed for applications and instances when a parameter changes to an OK state.
- false: events are generated and displayed for applications and instances when a parameter changes to an OK state.

---

**Note**

---

You must manually create this variable in your **patrol.conf** file.

---

---

# Support of Clusters and Failovers

This chapter discusses how the PATROL Agent supports an application in a cluster environment and what type of failover tolerance it provides. This chapter contains the following sections:

|  |      |
|--|------|
| Cluster and Failover Concepts .....                            | 6-3  |
| Control Script.....  | 6-3  |
| Cluster .....  | 6-3  |
| Cluster Node .....   | 6-3  |
| Cluster Application .....                                      | 6-3  |
| Cluster Management Software (CMS).....                         | 6-4  |
| Failover .....   | 6-4  |
| Group .....  | 6-4  |
| Package .....  | 6-5  |
| Virtual IP Address .....                                       | 6-5  |
| Examples of Third Party Cluster Software.....                  | 6-5  |
| PATROL Cluster Compatibility and Failover Tolerance .....      | 6-6  |
| Failover Behavior .....  | 6-7  |
| Description.....   | 6-7  |
| Illustration .....   | 6-9  |
| PATROL Agent Issues and Concerns.....                          | 6-12 |
| One Virtual IP Address For Each Group or Package.....          | 6-12 |
| PATROL Must Recognize the Virtual IP Address.....              | 6-12 |
| Local KM Customizations .....                                  | 6-13 |
| Basic Set Up of the PATROL Agent in a Windows Cluster.....     | 6-14 |
| Using the PATROL Cluster Configuration Wizard .....            | 6-14 |
| Manually Configuring the PATROL Agent for Clustering .....     | 6-15 |
| Define the PATROL Cluster-Specific Environment Variables... .. | 6-17 |
| Create and Register a New Service for the PATROL Agent .....   | 6-18 |

|  |      |
|--|------|
| Define the PATROL Agent as a Member of the Group . . . . .     | 6-20 |
| Basic Set Up of the PATROL Agent in a Unix Cluster. . . . .    | 6-23 |
| Configure PATROL Agent to Operate in a Cluster . . . . .       | 6-23 |
| Define the PATROL Cluster-Specific Environment Variables . . . | 6-25 |
| Define the PATROL Agent as a Member of the Package. . . . .    | 6-26 |
| PATROL Cluster-Specific Environment Variables for History and  |      |
| Configuration . . . . .  | 6-28 |
| Variables . . . . .  | 6-28 |
| Operation . . . . .  | 6-29 |
| Example . . . . .  | 6-30 |

# Cluster and Failover Concepts

This section defines some common terms and concepts that are used in the description of the PATROL Agent's support for cluster environments.

## Control Script

In a Unix environment, a control script governs the movement of a package from one host to another. It literally controls the failover process.

## Cluster

A cluster is a collection of two or more host computers that can connect and control common disk storage. Each cluster is controlled by a cluster management system (which can range from a control script on Unix to a third party application on Windows) that operates within the cluster.

## Cluster Node

Each host computer that shares physical disks or other storage devices and that are registered with the cluster management software (or control script) are referred to as cluster nodes.

## Cluster Application

A cluster application consists of all the necessary file structures that support the application. Cluster applications can run on any number of hosts that are logically grouped to form a cluster. They are commonly referred to as packages on Unix systems and groups on Window systems.

## Cluster Management Software (CMS)

Each cluster is controlled by cluster management software (CMS) operating within the cluster. The CMS controls the fail-over process. CMS ranges from third-party software applications to internally customized control scripts.

When the cluster application on a cluster node quits servicing requests, the CMS node shuts down the cluster application and disconnects (Windows) or unmounts (Unix) the physical disk storage that supports the cluster application. The CMS then designates another host to run the cluster application. The receiving host takes control of the physical disk storage and restarts the cluster application contained in the failover group (Windows) or package (Unix).

## Failover

Failover support is the ability to move a running cluster application from one cluster node to another with minimal data loss. The cause of the failover can range from scheduled maintenance to hardware or software failure.

## Group

A group is a logical grouping of application files and resources in a Windows environment. It consists of all necessary file structures required to run a cluster application and make it available to the end user. In the DNS server or other host name/IP Address resolution table, you must assign a virtual IP Address to the group. The cluster management software identifies the group by its virtual IP Address.

## Package

A package is a logical grouping of application files and resources in a Unix environment. It consists of all necessary file structures required to run a cluster application and make it available to the end user. In the DNS server or other host name/IP Address resolution table, you must assign a virtual IP Address to the package. The control script identifies the package by its virtual IP Address.

## Virtual IP Address

Each cluster application (also referred to as a package on Unix and a group on Windows) is assigned at least one IP address. This IP Address is used by the end-user front-end application for locating the application on the network. It is a virtual IP Address that is not associated with a physical location, but is associated with an application. The address is referred to as either “virtual” or “soft” because of its ability to be active on any host currently supporting the cluster application.

## Examples of Third Party Cluster Software

Some examples of third party clustering software include:

- MC ServiceGuard by Hewlett Packard
- High Availability Cluster Multi-Processing for AIX by IBM
- Sun Cluster the E6000 servers by Sun
- Windows Cluster Management Software

# PATROL Cluster Compatibility and Failover Tolerance

Failover tolerance is the ability to have an application that is running on one node in a cluster environment stop running on that node and have another node take over running the application. An application may quit running on a node for reasons such as:

- application failure
- hardware resource failure
- software resource failure
- load balancing software moves application
- administrator moves application

The PATROL Agent provides failover tolerance by

- monitoring all hosts in the cluster using the same configuration information
- recording the application history in the same history database for all hosts in the cluster

The PATROL Agent's failover tolerance prevents irregularities and gaps from occurring in the agent's history files for the cluster application. It saves you from having to manually reconcile two history files because an application failed on one host and another host took over, creating a separate set of history files.

The PATROL Agent uses three environment variables to provide failover support, including:

- `PATROL_CONFIG_PORT`—contains the fully qualified path to the configuration file stored on a drive shared to the cluster
- `PATROL_HISTORY_PORT`—contains the fully qualified path to the history file stored on a drive shared to the cluster
- `PATROL_VIRTUALNAME_PORT`—contains the virtual sever name that is used by the Patrol Agent instead of the hostname to store the PATROL configuration and historical data.

# Failover Behavior

The PATROL Agent supports failover in the manner described and illustrated here.

## Description

Table 6-1 describes the sequence of events that occur during a failover in a cluster environment and how the PATROL Agent behaves. This description uses Windows cluster terminology but also applies to Unix clusters.

**Table 6-1 PATROL Agent Behavior in a Cluster Environment (Part 1 of 2)**

|          | <b>Events</b>       | <b>Description</b>  | <b>Illustration</b> |
|----------|---------------------|---|---------------------|
| <b>1</b> | Request             | A user requests that the cluster application be started. A request can occur at anytime, including startup.   | Figure 6-1          |
| <b>2</b> | Start up            | The cluster management application runs the application on host A1. It starts the group which starts the cluster application and starts the PATROL Agent defined within the group. The agent reads the configuration file stored on a shared drive and starts up based on the file information. | Figure 6-1          |
| <b>3</b> | Run                 | The cluster application performs services and writes data to the shared disk drive. The agent monitors the application and writes history to the file specified by <i>PATROL_HISTORY_PORT</i> .   | Figure 6-1          |
| <b>4</b> | Service Interrupted | The cluster application is unable to access resources that it requires to run. The reason for this interruption can range from the failure of hardware or software to a forced failover by a system administrator.  | Figure 6-2          |
| <b>5</b> | Shutdown            | The cluster management software shuts down the cluster application and the PATROL Agent monitoring it on host A1.   | Figure 6-2          |

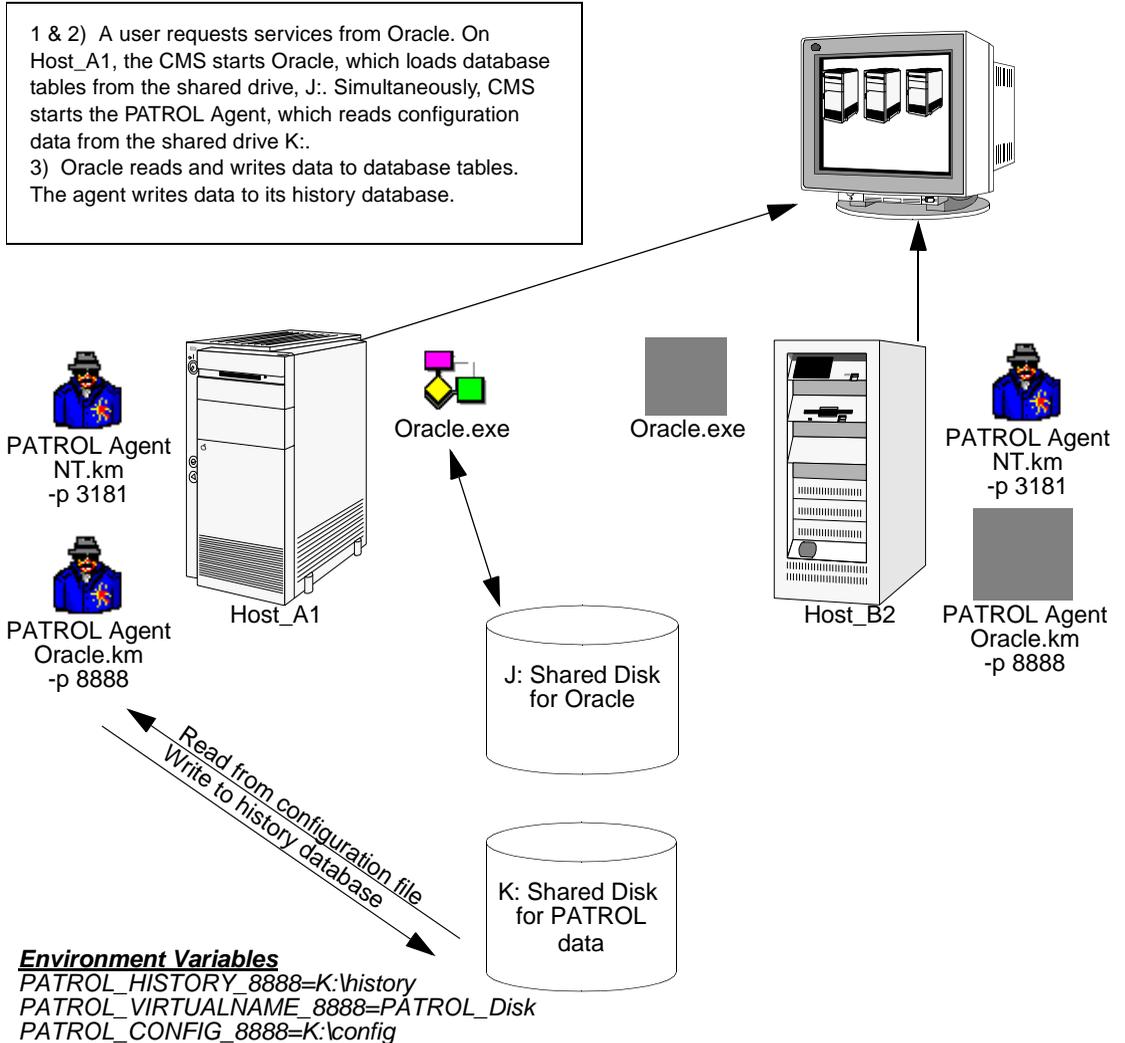
**Table 6-1 PATROL Agent Behavior in a Cluster Environment (Part 2 of 2)**

|          | <b>Events</b> | <b>Description</b>   | <b>Illustration</b> |
|----------|---------------|--|---------------------|
| <b>6</b> | Failover      | The cluster management software starts the group on B2, which starts the cluster application and the PATROL Agent.<br><br><b>Note:</b> Local KM customizations stored on A1 are not available for the PATROL Agent on B2.          | Figure 6-3          |
| <b>7</b> | Resume        | The cluster application performs services and writes data to the shared disk drive. The agent on host B2 monitors the application now running on B2 and writes history to the file specified by <code>PATROL_HISTORY_port</code> . | Figure 6-3          |

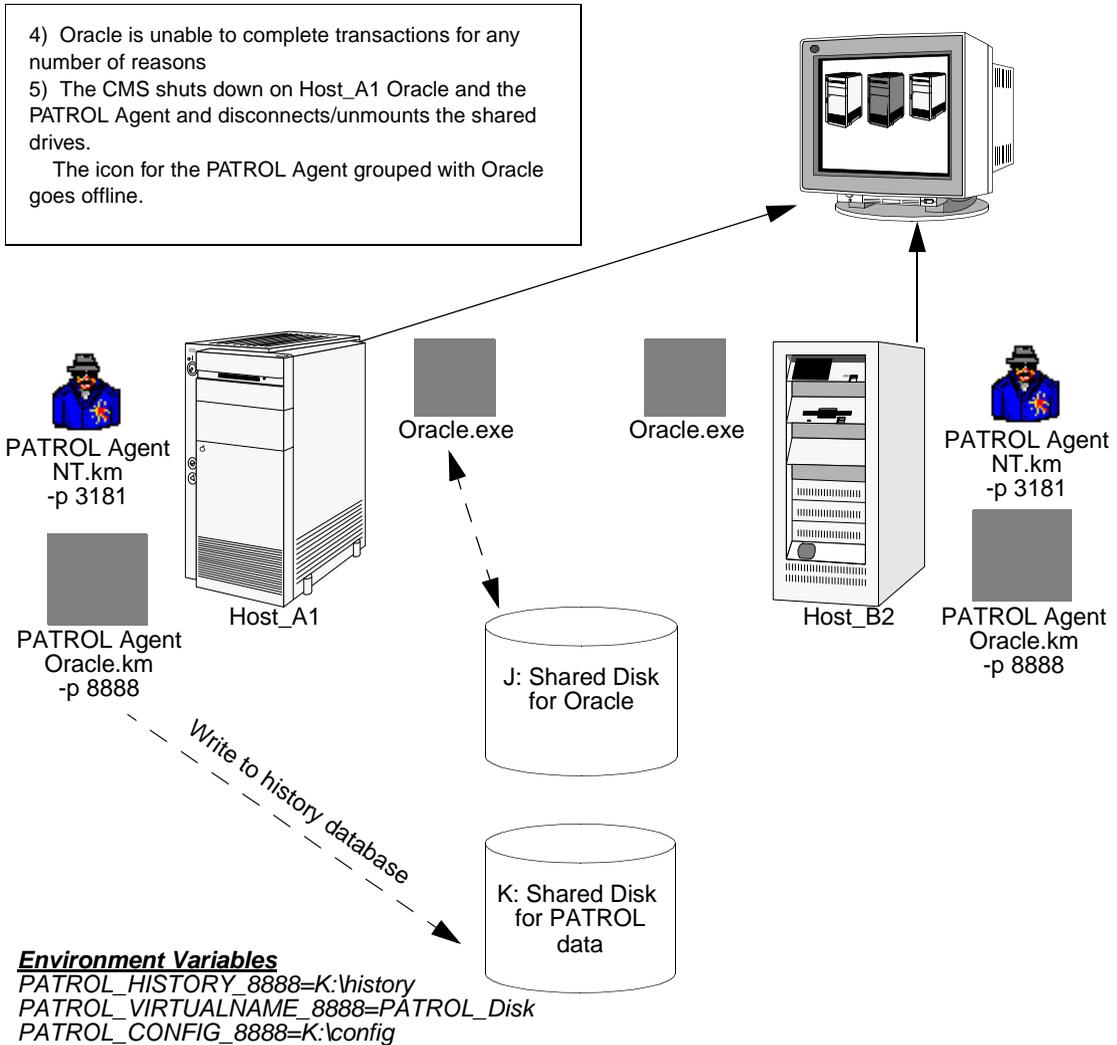
# Illustration

Figure 6-1, Figure 6-2, and Figure 6-3 graphically depict how the PATROL Agent behaves during a failover in a cluster environment.

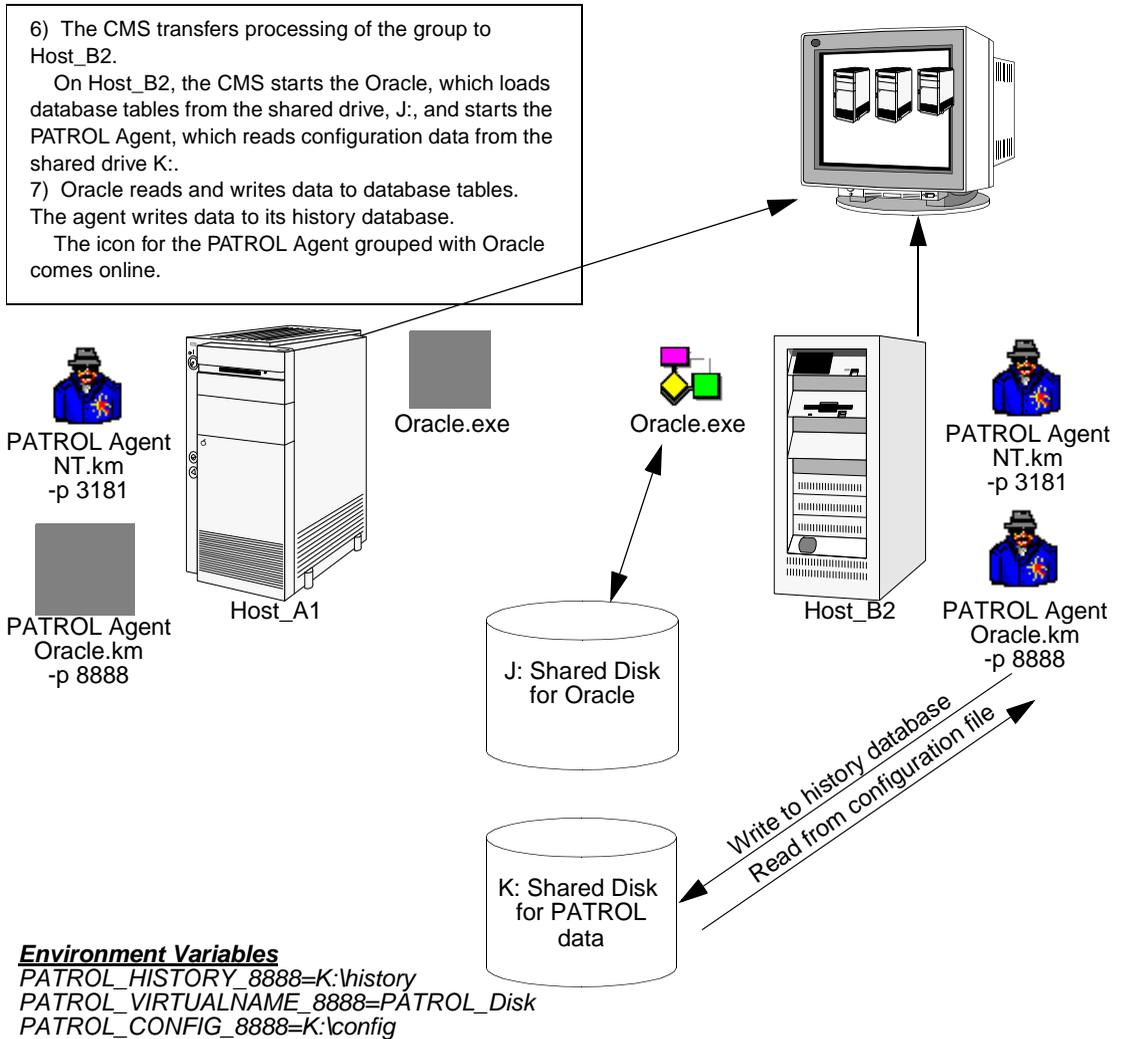
**Figure 6-1 Request, Start Up, and Run a Cluster Application**



**Figure 6-2 Experience Service Interruption and Shutdown of Cluster Application**



**Figure 6-3 Failover and Resumption of Cluster Application**



# PATROL Agent Issues and Concerns

The major concerns with a PATROL Agent in a cluster environment that supports failovers concern virtual IP addresses and local KM customizations.

## One Virtual IP Address For Each Group or Package

For PATROL Console applications and utilities to communicate with the agent after a group or package has failed over to another host, you must assign a virtual IP Address to the cluster application. A virtual IP address assigned to your DNS server or other host name/IP address resolution table guarantees that the console application can communicate with the agent regardless of which host it is running on.

## PATROL Must Recognize the Virtual IP Address

The cluster application's virtual IP address must be included in this list of IP addresses for the agent. This list is set in `/AgentSetup/IPAddresses`. PATROL does not accept communication packets from IP addresses that it does not recognize.

## Local KM Customizations

Local customizations are stored in the Knowledge Module with a host name designation. The only name that can be used is the actual host name of the PATROL Agent host. For this reason, it is important to use the actual host name when adding a PATROL Agent to the developer console. Using the actual host name ensures that the PATROL Agent loads the customizations.

Local customizations made using this method will not behave consistently when a failover occurs.

---

**Note**

---

BMC Software recommends that you make local customizations with an operator console with modified `patrol.conf` entry. These customizations are stored in the Agent Configuration File. For information on modifying `patrol.conf`, see *PATROL Console for Microsoft Windows 2000 User Guide, III—Customizing PATROL*.

---

# Basic Set Up of the PATROL Agent in a Windows Cluster

The information in this section provides a general idea of the processes involved in setting up a Windows cluster environment and integrating PATROL into that environment. Procedures and steps describing how to set up third-party software are intended as a general outline of the process for that product and are not intended as step-by-step instructions.

You can configure the PATROL Agent for your Windows cluster in the following ways:

- use the PATROL Cluster Configuration Wizard to automate the process of configuring the PATROL Agent into your Windows Cluster environment
- manually configure the PATROL Agent to work in your Windows Cluster environment

## Using the PATROL Cluster Configuration Wizard

The PATROL Cluster Configuration Wizard provides an easy-to-use interface that allows you to configure the PATROL Agent for failover in a Microsoft Cluster Server environment. While guiding you through the configuration process, the wizard collects the required configuration data and updates the system environment to integrate the PATROL Agent into the cluster.

If you configure your environment with the PATROL Cluster Configuration Wizard, you need only one PATROL Agent executable on your hard drive; however, you will have one PATROL Agent executable loaded into memory for every application you want to monitor on the cluster. To allow for using only one PATROL Agent executable, the PATROL Cluster Configuration Wizard replicates the original PATROL Agent entry in the Windows NT Registry and then modifies the display name and port number of the new PATROL Agent entries.

For additional information about the PATROL Cluster Configuration Wizard, see the *PATROL For Microsoft Windows Servers Getting Started*.

## Manually Configuring the PATROL Agent for Clustering

Setting up PATROL to run in a Windows cluster environment consists of several standard tasks. The standard cluster administration tasks and the PATROL-specific tasks are described in general terms. This section provides a high-level overview of building a Windows cluster and integrating PATROL into that environment.

The manual process defined in this chapter requires you to run multiple PATROL Agent executables on your CPU if you want to monitor more than one application on the cluster.

### Install the Application on each Cluster Node

Install the cluster application on the local disk. In the Windows environment, the executable must be installed on the local disk.

### Install the PATROL Agent on each Cluster Node

Install the PATROL Agent on the local disk of the node. You should have at least two separate agent executables installed on the node:

- one to monitor the node's operating system
- one to monitor the cluster application

Install the agent once. Include only those Knowledge Modules that support the application and the operating system. Then see “Create and Register a New Service for the PATROL Agent” on page 6-18 for information on setting up a second agent to monitor the cluster application.

## **Assign a Unique Port Number**

During installation of the agent on each node, assign a unique, listening port number to the PATROL Agent bound to the cluster application. This port must be the same across all nodes within the cluster.

## **Distribute License File**

Duplicate the license file on each node. Use the naming convention “license” without the host name as an extension. During startup, the PATROL Agent searches for “license.*hostname*,” using its own host name. If it can’t find the file, it searches for “license” without an extension.

If you duplicate a license file and do not delete or change the file’s host name extension, the agent cannot find the license and will not start.

## **Define the PATROL Cluster-Specific Environment Variables**

Create and set the PATROL Cluster-Specific variables. For more information, see “Define the PATROL Cluster-Specific Environment Variables” on page 6-17.

## **Create and Register a New Service for the PATROL Agent**

Set up a separate PATROL Agent process that can be bound to the cluster application. For more information, see “Create and Register a New Service for the PATROL Agent” on page 6-18.

## **Define the PATROL Agent as a Member of the Group**

Associate the PATROL Agent process with the cluster application. For more information, see “Define the PATROL Agent as a Member of the Group” on page 6-20.

## Define the PATROL Cluster-Specific Environment Variables

---

**Summary:** In this task, you will define the PATROL cluster-specific environment variables on each node. This action ensures that all the agents in a cluster read their configuration information and write their history information to the same set of files.

---

Perform the following tasks on each node in the cluster, then reboot each node. Rebooting enables each system to read the new variables and store them in memory.

- Step 1** From the Windows Taskbar, select **Start => Settings => Control Panel**.
- Step 1** Double-click the System icon and select the Environment tab.
- Step 1** Enter the variable name and value in the appropriate fields and click Set. The variables and their values are listed below. Repeat this step for the remaining variables.

```
PATROL_VIRTUALNAME_PORT=VirtualServerName  
PATROL_HISTORY_PORT=Drive:\History_Directory  
PATROL_CONFIG_PORT=Drive:\Config_Directory
```

For more information about specific variables, see “PATROL Cluster-Specific Environment Variables for History and Configuration” on page 6-28.

## Create and Register a New Service for the PATROL Agent

---

**Summary:** In this task, you will create a PATROL Agent executable and register it as a service so that you can dedicate it to monitoring a cluster application. This task involves copying and renaming the agent's executable and then registering the service in the Windows Services Applet.

---

Perform the following task on each node in the cluster.

- Step 1** Copy the PatrolAgent.exe in %PATROL\_HOME%\bin directory.
- Step 2** Rename the executable. Use a name that indicates that the agent is an executable dedicated to monitoring an application.
- Step 3** Paste the executable into the %PATROL\_HOME%\bin directory.

---

**Note**

---

Name the executable the same on every node in the cluster.

---

*PatrolAgent-application\_name.exe*

- Step 4** Install the executable at the command line, navigate to the %PATROL\_HOME%\bin directory, and enter the following command:

```
PatrolAgent-application_name -install
```

The system acknowledges that the service installed successfully.

```
Tue MON DD HH:MM:SS CCYY PatrolAgent-application_name  
PID 318 Success 1000:  
The PatrolAgent Service was successfully installed.
```

The PatrolAgent COM Server registered successfully

---

**Note**

---

The PatrolAgent COM Server can only be registered once. Additional attempts to register it will fail; however, the multiple agent processes will run.

---

- Step 5** From the Windows Taskbar, select **Start => Settings => Control Panel**.
- Step 6** Double-click the Services icon and select *application\_name* service from the list box. Click **Startup**.
- Step 7** In the Startup Type pane, select the Manual radio button and click OK. The service displays Manual in the Startup column.

## Define the PATROL Agent as a Member of the Group

---

**Summary:** In this task, you will add the new PatrolAgent service as a resource of type “Generic Service” to the cluster. This task is commonly referred to as binding the agent to the cluster application.

---

### Note

This task description uses Windows Cluster Management Software as an example. The steps describing how to set up the software are intended as a general outline of the process and are not intended as step-by-step instructions.

---

Perform the following task on only the master node of the cluster. The cluster software provides two methods for binding a service to a cluster: GUI or command line. Regardless of which method you choose, you must provide the information listed in Table 6-2.

**Table 6-2 Cluster Administration Properties (Part 1 of 2)**

| Arguments                       | Description  |
|---------------------------------|--|
| cluster.exe                     | Cluster Administration Executable (command line only)  |
| clusterName                     | User-defined name of the cluster   |
| RES                             | Specifies the service as a resource of the cluster   |
| "PatrolAgent for MyApplication" | Description of the service   |
| /CREATE /Group:<br>/TYPE:       | Create a group and assign it a resource type.  |
| /ADDEP                          | Establish a dependency between the service and the cluster.  |
| /Prop:RestartAction             | Determines what the cluster does (shut down, wait, etc.) if PATROL Agent service fails and is unable to restart. |
| /Priv: ServiceName              | Identify the service name of the PATROL Agent service bound to the cluster application.                          |

**Table 6-2 Cluster Administration Properties (Part 2 of 2)**

| <b>Arguments</b>            | <b>Description</b>   |
|-----------------------------|--|
| /Priv:<br>StartupParameters | Specify startup characteristics such as port number.             |
| /ON                         | Make the PATROL Agent service available (online) to the cluster. |

## Using Cluster Administration GUI

Add the new PatrolAgent service as a resource of type “Generic Service” to the cluster using the Cluster Administrator GUI.

## Using the Command Line

To bind a PATROL Agent service to the cluster application, you must issue a number of commands. Each command contains the name of the cluster registration executable, the name of the cluster, RES, description of the service, and various attributes.

---

### Note

---

For each command, you must reenter the name of the cluster executable, the name of the cluster, the resource option, and the service name.

---

- Step 1** Go to the command line.
- Step 2** Name the service, designate it as a resource of the cluster, create a group, and assign it a resource type of “Generic Service”.

```
cluster.exe clusterName RES "PatrolAgent for  
MyApplication" /CREATE /Group:MyGroup /TYPE:"Generic  
Service"
```

- Step 3** Add the disk that stores the PATROL Agent configuration and history information as a dependency. This command instructs the cluster software to bring up the disk with configuration information before it attempts to start the PATROL Agent.

```
cluster.exe clusterName RES "PatrolAgent for
MyApplication" /ADDDEP:"Disk MyGroupDisk"
```

- Step 4** Set the restart action. This command determines what the cluster does if an application fails and is unable to restart. A value of one (1) indicates that if the application is unable to restart, the cluster will continue to run.

```
cluster.exe clusterName RES "PatrolAgent for
MyApplication" /Prop:RestartAction=1
```

- Step 5** Identify the service name to the cluster software. The service name must be identical to the service name assigned to the PATROL Agent executable on each cluster node.

```
cluster.exe clusterName RES "PatrolAgent for
MyApplication" /Priv
ServiceName="PatrolAgent-application_name"
```

- Step 6** Set the port number for the PATROL Agent bound to the cluster application. This number must be the same as the number assigned as a suffix to the PATROL cluster-specific environment variables.

For details about the PATROL cluster-specific environment variables, see “Define the PATROL Cluster-Specific Environment Variables” on page 6-17.

```
cluster.exe clusterName RES "PatrolAgent for
MyApplication" /Priv StartupParameters="-p Port#"
```

- Step 7** Set the service to be available (online) when the cluster is running.

```
cluster.exe clusterName RES "PatrolAgent for
MyApplication" /ON
```

# Basic Set Up of the PATROL Agent in a Unix Cluster

The information in this section provides a general idea of the processes involved in setting up a Unix cluster environment and integrating PATROL into that environment. Procedures and steps describing how to set up third-party software are intended as a general outline of the process for that product and are not intended as step-by-step instructions.

## Configure PATROL Agent to Operate in a Cluster

Setting up PATROL to run in a Unix cluster environment consists of several standard tasks. The standard cluster administration tasks and the PATROL-specific tasks are described in general terms. This section provides a high-level overview of building a Unix cluster and integrating PATROL into that environment.

### Install the Application on a Shared Disk

Install the cluster application on a shared disk that's available to the entire cluster.

In Unix environments, the executable can be installed on the local disk. Such an installation strategy may call for the purchase of additional licenses and require additional administrative and maintenance tasks.

### Install the PATROL Agent on each Cluster Node

Use the same installation strategy for the PATROL Agent that you used for the cluster application.

---

**Note**

---

You should have at least one agent executable installed on the node's local drive to monitor the node's operating system. If you choose to install the cluster application on the local disk, you'll also have one to monitor the cluster application.

---

## Assign a Unique Port Number

During agent installation on each node, assign a unique, listening port number to the PATROL Agent bound to the cluster application. If you install the agent on each node, the port must be the same across all nodes within the cluster.

## Distribute License File

Duplicate the license file on each node. Use the naming convention “license” without the host name as an extension. During startup, the PATROL Agent searches for “license.*hostname*,” using its own host name. If it can’t find the file, it searches for “license” without an extension.

If you duplicate a license file with a host name extension and do not delete or change the file’s host name extension, the agent will only be able to find the license on the host from which the file originated. It will not start on the other hosts in the cluster.

## Define the PATROL Cluster-Specific Environment Variables

Create and set the PATROL Cluster-Specific variables. For more information, see “Define the PATROL Cluster-Specific Environment Variables” on page 6-25.

## Define the PATROL Agent as a Member of the Package

Associate the PATROL Agent process with the cluster application. For more information, see “Define the PATROL Agent as a Member of the Package” on page 6-26.

## Define the PATROL Cluster-Specific Environment Variables

---

**Summary:** In this task, you will define the PATROL cluster-specific environment variables in the `.profile` (Korn and Bourne shell) or `.cshrc` file (for C shell) of the account used to start the agent. This action ensures that the agent reads configuration information and writes history information to the same set of files, regardless of which host it is running on.

---

Perform the following task on each node in the cluster. Then, reboot each node. Rebooting enables each system to read the new variables and store them in memory.

»» For Korn and Bourne shells, use the following format:

```
PATROL_VIRTUALNAME_PORT=virtual server name
PATROL_HISTORY_PORT=Drive:\History_Directory
PATROL_CONFIG_PORT=Drive:\Config_Directory
```

```
export PATROL_VIRTUALNAME_PORT
export PATROL_HISTORY_PORT
export PATROL_CONFIG_PORT
```

»» For C shell, use the following format:

```
setenv PATROL_VIRTUALNAME_PORT virtual server name
setenv PATROL_HISTORY_PORT Drive:\History_Directory
setenv PATROL_CONFIG_PORT Drive:\Config_Directory
```

For more information about specific variables, see “PATROL Cluster-Specific Environment Variables for History and Configuration” on page 6-28.

## Define the PATROL Agent as a Member of the Package

---

**Summary:** In this task, you will add the PatrolAgent service to the cluster application's control script for a package. This task is commonly referred to as binding the agent to the cluster application.

---

---

### Note

---

The following code is a subset of the control script and does not represent a complete script. It is provided to give you an idea of the type and extent of changes you will need to make in HP's MC ServiceGuard or other Unix cluster management software.

---

» Make the following additions to the `customer_defined_run_cmds`:

```
# Start of user-defined functions.

function customer_defined_run_cmds
{
# Add customer defined run commands.
# PATROL Start command (substitute correct path
# and port).

    su patrol -c "/opt/PATROL3.4/PatrolAgent -p 3939"

# End of Patrol Start command.

    test_return 51
}
```

» Make the following additions to the `customer_defined_halt_cmds`:

```
function customer_defined_run_cmds
{
# Add customer defined halt commands.
# PATROL halt command (substitute correct path
# and port).

PID=$(ps -ef|awk '$NF=="3939" { print $2}')
```

```
if[[ -n $PID ]]
then
    kill -TERM $PID
fi

# End of Patrol Start command.

test_return 52
}
```

# PATROL Cluster-Specific Environment Variables for History and Configuration

To take advantage of failover tolerance for history files, you must create and set the value of three environment variables. When creating and writing to history files, the PATROL Agent searches for information in these files.

## Variables

Table 6-3 lists the PATROL cluster-specific environment variables and describes their purposes.

**Table 6-3 PATROL Cluster-Specific Environment Variables**

| Environment Variable                                       | Description   |
|--|---|
| PATROL_HISTORY<br>PATROL_HISTORY_PORT <sup>a</sup>         | the location of history files<br><br>If this variable is empty or doesn't exist, the agent writes the history files to <i>PATROL_HOME</i> \log\history\ <i>host</i> \ <i>portnumber</i> . |
| PATROL_VIRTUALNAME<br>PATROL_VIRTUALNAME_PORT <sup>a</sup> | an alias for the host name<br><br>If this variable is empty or doesn't exist, the agent uses the host name to identify history data within the history files.                             |
| PATROL_CONFIG<br>PATROL_CONFIG_PORT <sup>a</sup>           | the location of the configuration files<br><br>If this variable is empty or doesn't exist, the agent stores the configuration file in <i>PATROL_HOME</i> \config.                         |

<sup>a</sup> To manage multiple PATROL Agents running on separate ports, append the port number to the variable name. This situation occurs when individual PATROL Agents are bound to individual applications such as Oracle, Exchange, Sybase, etc. Each agent uses a separate port number.

# Operation

When searching for configuration information and creating and writing to the history database, the PATROL Agent uses the following logic to check for the existence of PATROL cluster-specific variables.

**Table 6-4 Operation of Configuration and History Environment Variables**

| Variable Type      | Exists? | Description   |
|--------------------|---------|---|
| Virtual Name       | yes     | <p>PATROL_VIRTUALNAME_8888 exists, the agent writes history using the virtual name as the host name. Using the virtual name provides continuous history for an application regardless of which host the application is running on.</p> <p>The agent also uses the virtual host name to identify the configuration file changes and the history database. Configuration file changes are written to <i>PATROL_HOME\config\config_virtualname_port.cfg</i>. The history database is written to the subdirectory structure <i>history\virtualname\port</i>, which will be located in the directory pointed to by <i>PATROL_HISTORY_PORT</i>.</p> |
|                    | no      | <p>The agent writes history using the actual host name. If the application fails over, the agent writes history using the new agent's name. Using the actual hostname creates gaps in the results of any <i>dump_hist</i> commands because the command does not recognize that the same application ran on different hosts.</p>   |
| Configuration File | yes     | <p>PATROL_CONFIG_8888 exists, then the agent reads configuration information from the location specified by this variable.</p>  |
|                    | no      | <p>The agent reads from the default directory, <i>PATROL_HOME\config\config_virtualname or hostname-port</i></p>  |
| History Database   | yes     | <p>PATROL_HISTORY_8888 exists, then the agent writes history to the location specified by this variable</p>   |
|                    | no      | <p>the agent writes to the default directory, <i>PATROL_HOME\log\history\virtualname or hostname\port</i></p>   |

## Example

The following example illustrates how the environment variables would be named for a host using port 8888. It also depicts the directory structure and file location.

### Environment Variables

```
PATROL_HISTORY=K:\doc\work\histdir  
PATROL_VIRTUALNAME=AliasHostName  
PATROL_CONFIG=K:\doc\work\config
```

### Directory Structure

For the values provided in the “Environment Variables” section of this example, the PATROL Agent stores configuration information and records the history data in the following directory structure:

```
K:\doc\work\histdir\AliasHostName\8888\annotate.dat  
K:\doc\work\histdir\AliasHostName\8888\param.hist  
K:\doc\work\config\config_AliasHostName-8888
```

If these variables do not exist or they are empty, the PATROL Agent stores configuration information and records the history data in the following directory structure:

```
%PATROL_HOME%\log\history\HostName\8888\annotate.dat  
%PATROL_HOME%\log\history\HostName\8888\param.hist  
%PATROL_HOME%\config\config_HostName-8888
```

---

# Loading and Monitoring Applications

This chapter describes how to manage the PATROL Agent as it monitors an application. The discussion includes:

- how to instruct the agent to load a knowledge Module
- how to begin monitoring when the agent starts up
- how to monitor when a console with the application loaded connects to the agent
- how to not monitor

This chapter also provides information on how to use agent configuration variables as filters to exclude or include certain applications and application instances in the following topics:

|   |      |
|---|------|
| Loading Knowledge Modules by Console and Agent . . . . .      | 7-3  |
| When the PATROL Agent Will Not Monitor Applications . . . . . | 7-3  |
| Version Arbitration. . . . .                                  | 7-4  |
| Application Status . . . . .                                  | 7-6  |
| Statuses . . . . .  | 7-6  |
| Assigning Status. . . . .                                     | 7-6  |
| Preloading Applications . . . . .                             | 7-7  |
| Based on Application Name . . . . .                           | 7-7  |
| Based on Architecture . . . . .                               | 7-8  |
| Designating Applications as Static. . . . .                   | 7-9  |
| Disabling Applications . . . . .                              | 7-9  |
| Based on Application Name . . . . .                           | 7-10 |

|   |      |
|---|------|
| Based on Architecture .....                                       | 7-11 |
| Filter Processing Logic For Disabling KMs .....                   | 7-12 |
| Types .....   | 7-12 |
| Precedence .....  | 7-13 |
| Example .....   | 7-14 |
| Listing Loaded Applications .....                                 | 7-16 |
| Running dump_km_list .....  | 7-16 |
| Example .....   | 7-17 |
| Selecting Which Instances to Monitor .....                        | 7-18 |
| Choosing an Inclusive or Exclusive Filter .....                   | 7-18 |
| Editing List to Filter by Application or Regular Expression . . . | 7-19 |

# Loading Knowledge Modules by Console and Agent

A PATROL Operator Console and a PATROL Developer Console both permit you to load and unload PATROL Knowledge Modules. If the PATROL Agent has another version of a PATROL Knowledge Module loaded, the PATROL Console uses its own version. The process through which the PATROL Console and PATROL Agent determine which version of a PATROL Knowledge Module to load is called version arbitration.

---

## Note

---

KM version arbitration does not apply to PATROL Central Operator – Microsoft Windows Edition or PATROL Central Operator – Web Edition.

---

## When the PATROL Agent Will Not Monitor Applications

The PATROL Agent will not monitor applications that you have loaded from the PATROL Console when the following three conditions exist:

- The PATROL Agent does not have a PATROL Knowledge Module that you loaded from a PATROL Operator Console.
- The application class is included in the PATROL Agent's knowledge module exclusion list. (The PATROL Knowledge Module is disabled.)
- The application class definition specifies that the applications are invalid for the PATROL Agent version and computer class.

## Version Arbitration

The process through which the PATROL Console and PATROL Agent determine which version of a KM to load is called version arbitration. Table 7-1, “PATROL Knowledge Module Version Arbitration,” on page 7-4 shows what version of a KM is loaded by the PATROL Agent under different conditions.

---

**Note**

---

KM version arbitration does not apply to PATROL Central Operator – Microsoft Windows Edition or PATROL Central Operator – Web Edition.

---

**Table 7-1 PATROL Knowledge Module Version Arbitration (Part 1 of 2)**

| Type of PATROL Console That Loads PATROL Knowledge Module | KM Installed in /knowledge Directory of the PATROL Agent | KM Preloaded on the PATROL Agent | KM in the Disabled List for the PATROL Agent | Result   |
|---|--|----------------------------------|--|--|
| PATROL Operator Console                                   | Yes  | No                               | No   | The PATROL Agent loads the local version of the KM into memory.                |
|   | Yes  | No                               | Yes  | The PATROL Agent ignores the PATROL Operator Console’s request to load the KM. |
|   | Yes  | Yes                              | No   | The PATROL Agent allows the PATROL Operator Console to use the KM.             |
|   | No   | No                               | No   | The PATROL Agent does not collect information for the KM.                      |
|   | No   | N/A                              | Yes  | The PATROL Agent ignores the PATROL Operator Console’s request to load the KM. |

**Table 7-1 PATROL Knowledge Module Version Arbitration (Part 2 of 2)**

| <b>Type of PATROL Console That Loads PATROL Knowledge Module</b> | <b>KM Installed in /knowledge Directory of the PATROL Agent</b> | <b>KM Preloaded on the PATROL Agent</b> | <b>KM in the Disabled List for the PATROL Agent</b> | <b>Result</b>   |
|--|---|---|---|---|
| PATROL Developer Console   | Yes   | No                                      | No  | The PATROL Agent loads into memory the most recent version of the KM either locally or from the PATROL Developer Console. |
|  | Yes   | No                                      | Yes   | The PATROL Agent ignores the PATROL Developer Console's request to load the KM.   |
|  | Yes   | Yes                                     | No  | The PATROL Agent loads the KM from the PATROL Developer Console into memory if the KM is newer.                           |
|  | No  | N/A                                     | No  | The PATROL Agent loads into memory the KM from the PATROL Developer Console.  |
|  | No  | N/A                                     | Yes   | The PATROL Agent ignores the PATROL Developer Console's request to load the KM.   |

For more information about KM version arbitration, refer to the *PATROL Console for Microsoft Windows 2000 User Guide, III—Customizing PATROL* or the *PATROL for Unix*.

# Application Status

You can control how, when, and if the PATROL Agent monitors applications by assigning each application a load status.

## Statuses

Table 7-2 lists the application loading statuses for the knowledge modules or knowledge module lists and the agent configuration variables used to assign each status. The Disabled and Static statuses cannot be applied to knowledge module lists (.kml).

**Table 7-2 Knowledge Module Statuses in the PATROL Agent**

| Status    | Description  | Agent Configuration Variable     |
|-----------|--|----------------------------------|
| Preloaded | The KM loads at startup and executes until the PATROL Agent stops.   | preloadedKMs<br>preloadedKMsArch |
| Disabled  | The KM will not load for any PATROL Console.   | disabledKMs<br>disabledKMsArch   |
| Static    | The KM does not load until a PATROL Console with the same KM connects to the PATROL Agent; when the console disconnects the KM remains loaded until the agent stops. This status ensures that the agent's history is continuous and complete. It provides constant monitoring without having to maintain a constant connection with a console. | staticApplications               |
| Dynamic   | The KM loads when a PATROL Console with the same KM connects to the PATROL Agent and unloads when the console disconnects (default).<br>If a knowledge module is not listed in one of the other variables, its loading status defaults to this value.  | none                             |

## Assigning Status

Assigning status to an application consists of entering the application's Knowledge Module file name, without the .km extension, in the appropriate agent configuration variable list.

# Preloading Applications

The PATROL Agent allows you to assign a preloaded status to an application. This status instructs the PATROL Agent to load an application's Knowledge Module and start monitoring it when the agent starts up.

PATROL allows you to preload applications

- by name—Knowledge Module file name, without the .km extension
- by architecture—operating system and platform

## Based on Application Name

The `/AgentSetup/preloadedKMs` configuration variable specifies which Knowledge Modules the PATROL Agent automatically loads for all operating systems and machine types when the agent starts up.

---

### Example

---

```
"/AgentSetup/preloadedKMs" =  
{REPLACE="ORACLE , SYBASE , INFORMIX" }
```

---

|                                |  |
|--------------------------------|--|
| <b>Format and Type of Data</b> | item_1,item_2,item_n (does not support wildcards), Knowledge Modules   |
| <b>Default Value</b>           | none<br><br>By default, the PATROL Agent does not load any of the application KMs; however, the ALL_COMPUTERS.km and the KM for the current machine type are always preloaded and always static. |
| <b>Minimum and Maximum</b>     | not applicable   |
| <b>Dependencies</b>            | none   |
| <b>Recommendation</b>          | The file names must be on the same line.   |

## Based on Architecture

The `/AgentSetup/preloadedKMsArch` configuration variable specifies which Knowledge Modules are preloaded for a specific operating system or machine type.

---

### Example

---

```
" /AgentSetup/preloadedKMsArch" =
{REPLACE="/NT.* , OS2.* /CPU , KERNEL/ " }
```

---

|                                       |  |
|---------------------------------------|--|
| <p><b>Format and Type of Data</b></p> | <p>This variable consists of two fields enclosed in forward slashes (/). Separate multiple entries with a comma (,).</p> <p><i>MachineList</i>—a comma-separated list of machine types or regular expressions</p> <p><i>KMList</i>—a case-sensitive, comma-separated list of KMs (without .km file extension) or regular expressions</p> <p><b>/MachineList/KMList/</b> — load only on listed machines and only listed KMs</p> <p><b>/MachineList!/KMList/</b> — load only on listed machines, load all except listed KMs</p> <p><b>!/MachineList/KMList/</b> — load all except the listed machines, load only listed KMs</p> <p><b>!/MachineList!/KMList/</b> — load all except listed machines, load all except listed KMs</p> <p>You can also use the following format:<br/> <b>/MachineList/KMList/ , /MachineList/KMList/</b>—load only on listed machines and only listed KMs</p> <p>Implicit matches (using !) have a lower precedence than explicit matches.</p> |
| <p><b>Default Value</b></p>           | <p>none</p>  |
| <p><b>Minimum and Maximum</b></p>     | <p>not applicable</p>  |
| <p><b>Dependencies</b></p>            | <p>none</p>  |
| <p><b>Recommendation</b></p>          | <p>none</p>  |

# Designating Applications as Static

The `/AgentSetup/staticApplications` configuration variable specifies a list of static applications. The applications in this comma-separated list are marked as static once they are loaded on the console. The static designation means that the KM will remain loaded in the agent's memory until the agent stops. Specify "all" if you want all dynamically loaded applications to be marked as being static.

|                                |  |
|--------------------------------|--|
| <b>Format and Type of Data</b> | item_1,item_2,item_n, Knowledge Modules  |
| <b>Default Value</b>           | The field contains an empty string " ". However, by default the PATROL Agent marks the <code>ALL_COMPUTERS</code> and common machine-type Knowledge Modules as static. |
| <b>Minimum and Maximum</b>     | not applicable   |
| <b>Dependencies</b>            | none   |
| <b>Recommendation</b>          | If you want an application to remain in agent memory once it is loaded, regardless of whether a console is connected, add it to this variable.                         |

## Disabling Applications

The PATROL Agent allows you to assign a disabled status to an application. This status prevents the PATROL Agent from loading that application's Knowledge Module and monitoring it.

PATROL allows you to disable applications

- by name—Knowledge Module file name, without .km extension
- by architecture—operating system and platform

---

### Note

---

Disabling an application even prevents the PATROL Developer Console from downloading the Knowledge Module into the memory of the agent.

---

## Based on Application Name

The `/AgentSetup/disabledKMs` configuration variable prevents Knowledge Modules from being loaded by the agent regardless of whether a console is a developer console.

---

### Example

---

If you want to disable everything that begins with NT, OS2, and VMS, use `NT.*`, `OS2.*`, and `VMS.*`, not `NT*.`, `OS2*.`, and `VMS*`. The second type of filter will not produce the desired result.

```
"/AgentSetup/disabledKMs" = {REPLACE="NT.* , OS2.* ,  
VMS.* , bad_APPL" }
```

---

|                                |  |
|--------------------------------|--|
| <b>Format and Type of Data</b> | item_1,item_2,item_n, Knowledge Modules  |
| <b>Default Value</b>           | none   |
| <b>Minimum \ Maximum</b>       | not applicable   |
| <b>Dependencies</b>            | none   |
| <b>Recommendation</b>          | The ALL_COMPUTERS.km and the KM for the current machine type cannot be disabled.<br><br>The file names must be on the same line. |

## Based on Architecture

The `/AgentSetup/disabledKMsArch` configuration variable specifies which Knowledge Modules are prevented from being loaded for all specified operating systems and machine types.

---

### Example

---

```
"/AgentSetup/disabledKMsArch" =  
{REPLACE=/NT.* , OS2.* /CPU , KERNEL/ }
```

---

|                                |   |
|--------------------------------|---|
| <b>Format and Type of Data</b> | This variable uses the same format as the <code>/AgentSetup/preloadKMsArch</code> . For variable format information, see the format section of “Based on Architecture” on page 7-8. |
| <b>Default Value</b>           | none  |
| <b>Minimum and Maximum</b>     | not applicable  |
| <b>Dependencies</b>            | none  |
| <b>Recommendation</b>          | none  |

# Filter Processing Logic For Disabling KMs

The PATROL Agent builds filters to determine which applications it should disable. The agent builds these filters out of regular expressions and literal strings.

## Types

The PATROL Agent Configuration Variables `disabledKMs` and `disabledKMsArch` support three types of filters

- **Positive**—is inclusive; the filter selects only those entities that match an item in the list
- **Negative**—is exclusive; the filter rejects those items that match an item in the list
- **Implicit**—is associative; the filter selects or rejects items because the group or subset that the items belong to is selected or rejected; it consists of the opposite of the explicit. For example, if in the `disabledKMs` variable you explicitly enter `!NT/NT_CPU`, which translates to “do not disable NT\_CPU on NT systems,” you implicitly state disable NT\_CPU on all other systems, such as HP, AIX, and OpenVMS.

## Precedence

The agent processes the filters in the following order:

- 1 Positive
- 2 Negative
- 3 Implicit

When the agent finds a match based on a filter's specification, the match takes precedence over filters with lower precedence, except in the case of wildcard matches.

---

### Note

---

Exact matches of a lower filter order can override wildcard matches of a higher filter order.

---

## Entry order

The entry order of the clauses, separated by commas, does not affect the precedence of the filters.

## Wildcard vs. Exact Matches

An exact match takes precedence over a wildcard match and can override the filter precedence. An exact match in a second or third order filter overrides a wildcard match in a first order filter.

---

### Example

---

The disabledKMsArch variable contains the following values:

```
/NT/NT_*/, /NT/!NT_MEMORY,NT_CACHE/
```

Based upon this entry, all NT KMs are disabled except NT\_MEMORY and NT\_CACHE because the exact match in the negative filter clause takes precedence over the wildcard match in the positive filter.

---

## Exact Matches in Two or More Filters

If an item matches exactly in the positive and negative filters, the positive filter has precedence.

---

### Example

---

The disabledKMsArch variable contains the following values:

```
/NT/NT_FILESYSTEM, .* / ,  
/NT/!NT_CPU,NT_CACHE,NT_FILESYSTEM/ , /!AIX/NT_FTP/
```

The NT\_FILESYSTEM application would be disabled (not loaded) because the positive filter has precedence.

---

## Example

In the disabledKMsArch agent configuration variable, enter the following list separated by commas:

```
/NT/.*/ , /VMS/.* ,  
/NT/!NT_CPU,NT_MEMORY/ , /!VMS/VMS_QUEUE.* , VMS_PROCESS.* /
```

## Filters

The Windows agent builds three filters:

Positive Filter (/NT././AIX./VMS./) — disables (does not load) all KMs on all systems Windows NT and OpenVMS systems

Negative Filter (/NT/!NT\_CPU,NT\_MEMORY/) — does not disable (loads) CPU and memory applications on Windows NT systems

Implicit Positive Filter (/!VMS/VMS\_QUEUE.\*, VMS\_PROCESS.\*/) — explicitly enables (loads) the VMS queue and process applications on OpenVMS systems, which implies that these applications are disabled (not loaded) on Windows NT systems

## Results

For all monitored systems, the PATROL Agent loads the CPU and memory applications on all Windows NT systems. The agent also loads all queue and process applications on all OpenVMS systems. When the agent loads these applications depends upon each application's status (preload, static, and dynamic).

# Listing Loaded Applications

To view a list of applications currently loaded in the PATROL Agent's memory, run the `dump_km_list` utility.

The `dump_km_list` utility provides the following information for each application loaded in the agent's memory:

- application name
- application version
- application status (static or not)
- number of consoles attached to the agent that have the application loaded

## Running `dump_km_list`

- » Type the following command at the PATROL Console system output window and press **Enter**:

```
%DUMP KM_LIST
```

---

**Note**

---

This command is case-sensitive.

---

## Example

The following example is from a dump km\_list.

```
CCYY0509092435 === End of loaded KM list (44 entries)=====
CCYY0509092547 === Loaded Knowledge Modules (Applications)=====
CCYY0509092547 # Name                               Version Static (# of
                                                    consoles)
CCYY0509092547-----
CCYY0509092547 1: ALL_COMPUTERS                          11.0   yes   (3)
CCYY0509092547 2: NT                                           1.109  yes   (2)
CCYY0509092547 3: PATROL_NT                               10.20  no    (3)
CCYY0509092547 4: NT_SYSTEM                            1.49   no    (3)
CCYY0509092547 5: NT_SERVER                          1.39   no    (2)
...
CCYY0509092548 38: NT_Composites                       1.5    no    (1)
CCYY0509092548 39: NT_CompositesColl                          1.4    no    (1)
CCYY0509092548 40: NT_REGISTRY                       1.45   no    (2)
CCYY0509092548 41: NT_REGISTRY_GROUP                  1.37   no    (2)
CCYY0509092548 42: NT_REGISTRY_KEYINST                1.17   no    (2)
CCYY0509092548 === End of loaded KM list (42 entries) =====
```

---

### Warning

---

An operator console and a developer console are connected to the same agent. The operator console then loads KM that is not loaded on the developer console. When you run the dump km\_list utility on either of the consoles, the KM, which was loaded on the operator console but is not loaded on the developer console, is displayed in the list as version 0.0.

---

The scenario described above does not apply to PATROL Central Operator – Microsoft Windows Edition or PATROL Central Operator – Web Edition.

# Selecting Which Instances to Monitor

The PATROL Agent allows you to monitor an application while allowing you to include and exclude certain instances. The ability to filter instances in or out gives you the flexibility to monitor exactly what you want. This filtering feature permits you to enter the exceptions rather than create exhaustive lists of instances that constitute the rule.

## Choosing an Inclusive or Exclusive Filter

The `/AgentSetup/application_name.filterType` configuration variable determines the type of filter for a corresponding filter list. You must create this variable.

---

### Example

---

To exclude three instances of an Oracle database, you must create a filter type variable, `/AgentSetup/ORACLE.filterType = {REPLACE="exclude"}`, and a corresponding filter list variable, `/AgentSetup/ORACLE.filterList = {REPLACE="ORACLE_A,ORACLE_B,ORACLE_C"}`.

---

|                          |  |
|--------------------------|--|
| <b>Values</b>            | exclude—if discovered, do not monitor<br>include—if discovered, monitor  |
| <b>Default Value</b>     | exclude  |
| <b>Minimum \ Maximum</b> | not applicable   |
| <b>Dependencies</b>      | <code>/AgentSetup/application_name.filterList</code> specifies items (application instances) that are monitored (include) or not (exclude). Each <code>.filterList</code> needs a <code>.filterType</code> . |
| <b>Recommendation</b>    | Create a configuration variable of this type with a different application name for each application.<br><br>Valid application names are described in the Knowledge Module reference manuals.                 |

# Editing List to Filter by Application or Regular Expression

The `/AgentSetup/application_name.filterList` configuration variable lets you specify the set of application instances or regular expressions to include or exclude from monitoring, depending upon the type of filter.

---

### Example

---

```
"/AgentSetup/FILESYSTEM.filterList"  
={REPLACE="root,us.*,ora.*"}
```

---

|                                |   |
|--------------------------------|---|
| <b>Format and Type of Data</b> | item_1,item_2,item_n, application instances or regular expressions  |
| <b>Default Value</b>           | none  |
| <b>Minimum and Maximum</b>     | not applicable  |
| <b>Dependencies</b>            | <code>/AgentSetup/application_name.filterType</code> determines whether the items listed by this variable are monitored (included) or not (excluded).   |
| <b>Recommendation</b>          | Create a configuration variable of this type with a different application name for each application. Each <code>.filterList</code> needs a <code>.filterType</code> .<br><br>Valid application names are described in the Knowledge Module reference manuals. |



---

# Using pconfig to Configure the PATROL Agent

This chapter discusses configuring the PATROL Agent from the command line, the syntax of the `pconfig` utility, and the options to use for the task you want to perform. It also provides examples of configuring the PATROL Agent from the command line using the `pconfig` utility. This chapter contains the following sections:

|  |      |
|--|------|
| Overview of the pconfig Command Line Configuration Utility . . . . | 8-2  |
| Prerequisites for Configuring at the Command Line . . . . .        | 8-2  |
| Syntax of pconfig . . . . .  | 8-3  |
| Specifying a Windows Domain Account Name . . . . .                 | 8-5  |
| Determining the Command String to Use for the Task . . . . .       | 8-5  |
| Determining the Option to Use for the Task. . . . .                | 8-9  |
| Examples of Configuring at the Command Line . . . . .              | 8-10 |

# Overview of the pconfig Command Line Configuration Utility

To configure the PATROL Agent at the command line, use the `pconfig` utility. The `pconfig` utility allows you to manipulate the configuration files used by the PATROL Agent. The utility also allows you to specify which change file is applied to the configuration.

## Prerequisites for Configuring at the Command Line

Before you configure the PATROL Agent at the command line, make sure that you have performed these tasks:

- You have reviewed and understood Chapter 3, “Background About Configuring the PATROL Agent.”
- You know the change files you want to use (if any).
- You have manually changed the variables that you needed to change in the change files (if any).
- You have started the PATROL Agent. For more information on starting the agent, see Chapter 2, “Starting and Stopping the PATROL Agent.”
- For Windows, you have set the variable called `%PATROL_HOME%`.
- For Unix, you have run one of the following scripts from the PATROL installation directory to set the variable called:

```
> . ./patrolrc.sh (for Korn and Bourne shell)
> source .patrolrc (for C shell)
```

# Syntax of pconfig

The format of the pconfig utility is as follows:

```
pconfig+get [options]-save filename
```

[options] available for +get

```
-host hostname  
-debug  
-lp  
-port portnumber  
+tcp  
+verbose
```

```
pconfig-help
```

```
pconfig+KILL [options]
```

[options] available for +KILL:

```
-host hostname  
-debug  
-lp  
-port portnumber  
+tcp  
+verbose
```

```
pconfig+license [options]filename
```

[options] available for +license

```
-host hostname  
-debug  
-lp  
-port portnumber  
+tcp  
+verbose
```

```
pconfig[options] +PURGE [soptions][filename]
```

[options] available for +PURGE

```
-host hostname  
-debug  
-lp  
-port portnumber  
+tcp  
+verbose
```

[soptions] available for +PURGE

```
+Reload
```

```
pconfig[options] +Reload [soptions][filename]
```

[options] available for +Reload

```
-host hostname  
-debug  
-lp  
-port portnumber  
+tcp  
+verbose
```

[soptions] available for +Reload

```
+PURGE
```

```
pconfig+RESTART [options]
```

[options] available for +RESTART

```
-host hostname  
-debug  
-lp  
-port portnumber  
+tcp  
+verbose
```

```
pconfig -version
```

## Specifying a Windows Domain Account Name

When you want to specify a Windows domain account in the access subagentcontrol list (ACL), be sure to use two backslashes (\\) between the domain and the user name as shown in the following example. This operation is performed on the command line of the pconfig utility.

```
domain\\username/*/*
```

The pconfig utility will not accept the Windows domain notation without the backslashes.

## Determining the Command String to Use for the Task

Use Table 8-1 to find the `pconfig` command string for the task you want to perform. For information about the available options, see “Determining the Option to Use for the Task” on page 8-9.

**Table 8-1 pconfig Command Strings (Part 1 of 3)**

| Task to Perform  | Command String to Use   | Additional Information  |
|--|---|---|
| <p>Get a PATROL Agent's configuration and save it to a file.</p> | <pre>pconfig +get [options] -save filename  [options] available for +get -host hostname -debug -lp -port portnumber +tcp +verbose</pre> | <p>The resultant file contains the change information only. That is, this file can be used to re-create the source configuration when applied with the <code>config.default</code> file. The default is <code>stdout</code>.</p> <p><b>Note</b> – If the file already exists, the save option appends the new information to the end of the file. When using the host option, you can specify multiple host names. Information on multiple hosts is saved to the same file.</p> |
| <p>Get Help on the options for <code>pconfig</code>.</p>         | <pre>pconfig -help</pre>  | <p>This provides a brief description of the available options.</p>  |
| <p>Stop the PATROL Agent.</p>                                    | <pre>pconfig +KILL [options]  [options] available for +KILL: -host hostname -debug -lp -port portnumber +tcp +verbose</pre>             | <p>This command can be used to stop any agent that is accessible on the network.</p>  |

**Table 8-1 pconfig Command Strings (Part 2 of 3)**

| Task to Perform   | Command String to Use   | Additional Information  |
|---|---|---|
| <p>Reload the <code>config.default</code> if it has been changed.</p>   | <pre>pconfig [options] +Reload [+PURGE] filename</pre> <p>[options] available for +Reload</p> <ul style="list-style-type: none"> <li>-host <i>hostname</i></li> <li>-debug</li> <li>-lp</li> <li>-port <i>portnumber</i></li> <li>+tcp</li> <li>+verbose</li> </ul> | <p>This option does not remove or modify any changes applied to the configuration; it reapplies the underlying default configuration file and combines it with the changes. The result is a configuration database built from the new default configuration and from the changes that had been applied previously.</p> <p>You must specify the files to use to create the new configuration. You can use more than one file name separated by a space, or you can use <code>--</code> for <code>stdin</code>.</p> |
| <p>Remove the existing configuration from the destination agent and create another one from the <code>config.default</code> file.</p> | <pre>pconfig [options] +PURGE [+Reload filename]</pre> <p>[options] available for +PURGE</p> <ul style="list-style-type: none"> <li>-host <i>hostname</i></li> <li>-debug</li> <li>-lp</li> <li>-port <i>portnumber</i></li> <li>+tcp</li> <li>+verbose</li> </ul>  | <p>You can save configuration changes in a <code>.cfg</code> file.</p>  |
| <p>Restart the PATROL Agent.</p>  | <pre>pconfig +RESTART [options]</pre> <p>[options] available for +RESTART</p> <ul style="list-style-type: none"> <li>-host <i>hostname</i></li> <li>-debug</li> <li>-lp</li> <li>-port <i>portnumber</i></li> <li>+tcp</li> <li>+verbose</li> </ul>                 |   |

**Table 8-1** pconfig Command Strings (Part 3 of 3)

| <b>Task to Perform</b>                      | <b>Command String to Use</b>  | <b>Additional Information</b>                            |
|---|---|--|
| Send a license file to the PATROL Agent.    | <pre>pconfig +license [options] filename  [options] available for +license -host <i>hostname</i> -debug -lp -port <i>portnumber</i> +tcp +verbose</pre> |  |
| Display the version information of pconfig. | <pre>-version</pre>   | This option provides the version of the pconfig utility. |

# Determining the Option to Use for the Task

Use the table to find the `pconfig` option for the task you want to perform.

**Table 8-2** `pconfig` Options (Part 1 of 2)

| <b>Task to Perform</b>  | <b>Option to Use<sup>a</sup></b>            | <b>Additional Information</b>  |
|---|---|--|
| Get Help on the options for <code>pconfig</code> .                | -help                                       | You can use the -help option with another option to get a description of that option.  |
| Print error and status information to stdout as the utility runs. | +verbose<br>+v                              |  |
| Set the debugging level for <code>pconfig</code> .                | -debug                                      | The default is 0.  |
| Specify the configuration change files to be used.                | <i>filename1</i><br>[ <i>filename2...</i> ] | You must separate the names of change files with a blank space. This option cannot be used with the +get option. The default is <code>stdin</code> .   |
| Specify the host name of the agents.                              | -host <i>name</i><br>-ho                    | <p>You can specify either the host name for the agent from which you want to get a configuration or the host name for the agents you want to configure. The default is the name of the local host.</p> <p>You can specify multiple hosts using a comma-separated list with no intervening space. For more information, see “Examples of Configuring at the Command Line” on page 8-10.</p> |
| Specify the local port number of the agents.                      | -lp <i>num</i>                              | You can specify either the local port for the agent from which you want to get a configuration or the local port for the agents you want to configure. The default local port number is 0.   |
| Specify the port number of the agents.                            | -port <i>num</i><br>-p                      | You can specify either the port for the agent from which you want to get a configuration or the port for the agents you want to configure. The default port number for version 3.4 or later is 3181. The default port number for version 3.3 or earlier was 1987.  |
| Send a license file to a PATROL Agent.                            | +license<br>+l                              |  |

**Table 8-2** pconfig Options (Part 2 of 2)

| Task to Perform   | Option to Use <sup>a</sup> | Additional Information  |
|---|----------------------------|---|
| Specify that the changes come from <code>stdin</code> .   | --                         | This option can be used in place of <code>file1 [file2...]</code> .   |
| Display the version information of <code>pconfig</code> . | -version                   | This option provides the version of the <code>pconfig</code> utility. |

<sup>a</sup> Shortcut entries, where available, are listed below the option.

## Examples of Configuring at the Command Line

The following command gets the configuration changes (not the default values) from the PATROL Agent on port 3160, host `hudson`. The change information is saved to the `hud.cfg` file in the `config` directory.

```
pconfig +g -p 3160 -ho hudson -s /config/hud.cfg
```

The following command purges the configurations in the agents on port 3107 for hosts called `nile`, `ohio`, and `monongahela`. It then builds new configuration files from the `config.default` and the `hud.cfg` files.

```
pconfig +P -p 3107 -ho nile,ohio,monongahela  
/config/hud.cfg
```

# Using xpcnfig (Unix) to Configure the PATROL Agent

This chapter includes tasks for configuring the PATROL Agent with the xpcnfig configuration utility. This chapter contains the following sections:

|  |      |
|--|------|
| Overview of the xpcnfig Configuration Utility . . . . .                            | 9-3  |
| Prerequisites for Configuring with the xpcnfig Utility. . . . .                    | 9-4  |
| Overview of xpcnfig Functions . . . . .  | 9-5  |
| The xpcnfig Primary Window . . . . .   | 9-7  |
| Accessing the xpcnfig Primary Window . . . . .                                     | 9-8  |
| Closing the xpcnfig Primary Window . . . . .                                       | 9-10 |
| Cancelling Changes Not Applied. . . . .  | 9-10 |
| Stopping an Operation . . . . .  | 9-10 |
| Returning to the xpcnfig Primary Window . . . . .                                  | 9-10 |
| Selecting a Change File . . . . .  | 9-12 |
| When Changes Take Effect . . . . .   | 9-12 |
| Reinitializing the PATROL Agent . . . . .  | 9-13 |
| Reloading the config.default File and Change File for the<br>PATROL Agent. . . . . | 9-15 |
| Stopping the PATROL Agent. . . . .   | 9-17 |
| Handling Change Files . . . . .  | 9-18 |
| Tasks Available for Handling Change Files. . . . .                                 | 9-18 |
| Opening an Existing Change File . . . . .  | 9-19 |
| Creating a New Change File . . . . .   | 9-21 |
| Sending a Complete Set of Variables. . . . .                                       | 9-22 |
| Viewing a Change File. . . . .   | 9-23 |
| Saving a Change File . . . . .   | 9-24 |

|  |      |
|--|------|
| Applying the Changes to a PATROL Agent . . . . .                       | 9-26 |
| Purging the Existing Configuration from a PATROL Agent . . . . .       | 9-28 |
| Sending a New License File to a PATROL Agent . . . . .                 | 9-30 |
| Handling Variables . . . . .   | 9-32 |
| Handling Variables from the Primary Window . . . . .                   | 9-32 |
| Adding a Host to the Host List . . . . .                               | 9-33 |
| Adding New Variables to the Change File . . . . .                      | 9-36 |
| Deleting a Variable from the Change File . . . . .                     | 9-38 |
| Resetting a Variable to Its Default Value . . . . .                    | 9-40 |
| Handling Variables from the Edit Variable Dialog Box . . . . .         | 9-40 |
| Modifying Variables Using the Edit Variable Dialog Box . . . . .       | 9-41 |
| Inserting a New Change Entry in the Edit Variable Dialog Box . . . . . | 9-44 |
| Modifying a Change Entry in the Edit Variable Dialog Box . . . . .     | 9-46 |
| Deleting a Change Entry from the Edit Variable Dialog Box . . . . .    | 9-48 |
| Modifying the Default Account Variable in the Change File . . . . .    | 9-49 |

# Overview of the xpcnfig Configuration Utility

To configure the PATROL Agent on Unix, you can use the xpcnfig configuration utility. The xpcnfig configuration utility allows you to manipulate both the configuration files used by the PATROL Agent and the variables in those files.

You can perform the following tasks with the xpcnfig utility:

- cancel changes
- stop an operation
- return to the primary window
- select a change file
- apply changes
- reinitialize (restart) the PATROL Agent
- reload the **config.default** file and the change file
- stop the PATROL Agent
- start the xpcnfig configuration utility
- exit the xpcnfig configuration utility

# Prerequisites for Configuring with the xconfig Utility

Before you configure the PATROL Agent with the xconfig configuration utility, make sure you have completed these prerequisites:

- You have reviewed and understood Chapter 3, “Background About Configuring the PATROL Agent.”
- You know the change files you want to use or the variables you want to add, modify, or delete.
- You have started the PATROL Agent. (See Chapter 2, “Starting and Stopping the PATROL Agent.”) Starting the agent is only necessary to apply changes to the agent or to get configuration changes from it.
- If starting xconfig from a command line, you have set display variables.

If you want to change values for variables, make sure you have completed these prerequisites:

- You have write permission on the Agent Configuration Change File.
- You know the variables that you need to change in the change files.

# Overview of xpcnfig Functions

Table 9-1 provides an overview of the menus and menu commands included in xpcnfig.

**Table 9-1** Menus and Menu Commands for the xpcnfig Utility (Part 1 of 2)

| <b>Menu</b> | <b>Command</b>   | <b>Description</b>   |
|-------------|------------------|--|
| File        | Open...          | Creates a new PATROL Agent configuration by loading the <b>config.default</b> file.            |
|             | Save             | Saves the change file for the PATROL Agent.  |
|             | Save As...       | Saves the change file for the PATROL Agent to a name you specify.                              |
|             | Exit             | Closes the xpcnfig window.   |
| Edit        | Add...           | Displays the Add Variable dialog box to insert a new variable into the configuration.          |
|             | Modify...        | Displays the Modify Variable dialog box to modify an existing variable in the configuration.   |
|             | Reset to Default | Resets the selected variable to its default value.   |
|             | Delete           | Displays the Delete Variable dialog box to delete an existing variable from the configuration. |
|             | Select All       | Selects all variables in the configuration.  |
|             | Deselect All     | Deselects all variables in the configuration.  |
|             | Delete All       | Deletes all variables in the configuration.  |
| Host        | Add...           | Adds a host that is not on the Hosts box.  |
|             | Delete           | Removes the selected host from the Hosts box.  |
|             | Select All       | Selects all the host names in the Hosts box.   |
|             | Deselect All     | Deselects all the host names in the Hosts box.   |
|             | Delete All       | Deletes all the host names in the Hosts box.   |

**Table 9-1 Menus and Menu Commands for the xpcnfig Utility (Part 2 of 2)**

| <b>Menu</b> | <b>Command</b>     | <b>Description</b>  |
|-------------|--------------------|---|
| Options     | View...            | Use this command to view the contents of a change file. You can search for entries while viewing the file.  |
|             | Reload             | Reloads the <b>config.default</b> file and the change file for a PATROL Agent.  |
|             | Purge              | Removes the existing configuration from a PATROL Agent and builds a new configuration from the <b>config.default</b> file and the change file for the PATROL Agent. |
|             | Reinitialize Agent | Displays the Reinitialize Agent dialog box to restart the PATROL Agent.   |
|             | Stop Agent         | Displays the Kill Agent dialog box to stop the PATROL Agent.  |
|             | License...         | Use this procedure to send a new license file to a PATROL Agent on selected hosts.  |
|             | Send Defaults      | Use this procedure to send a complete set of variables to the PATROL Agents to use as the defaults.   |

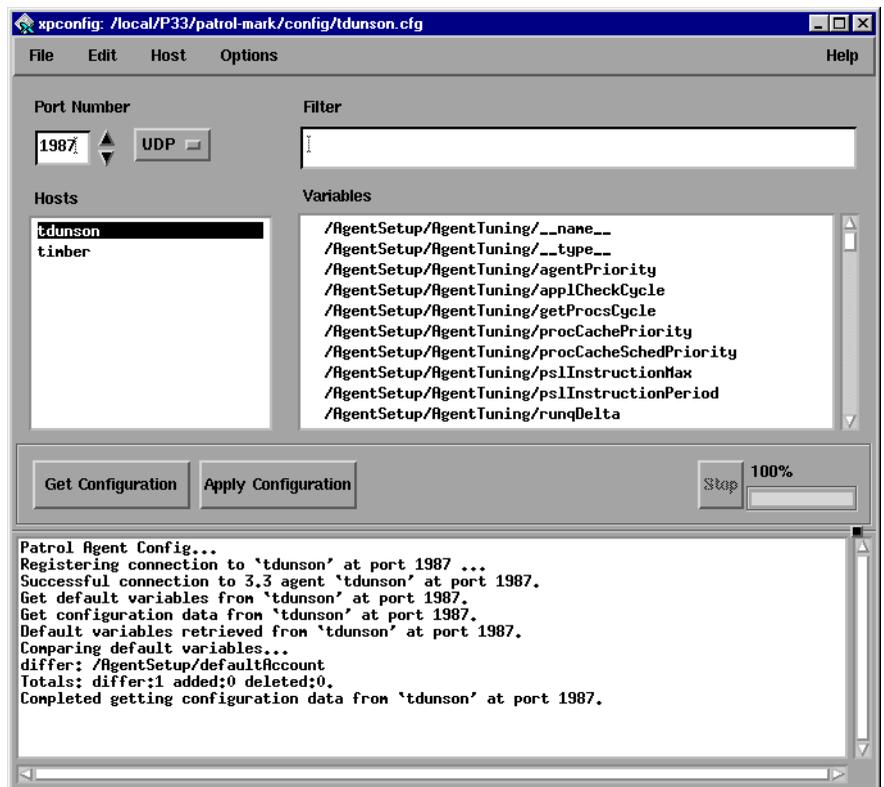
# The xpcnfig Primary Window

The xpcnfig window displays the current configuration. Highlighting a host name on the left causes the Variables box on the right to display the name and path of each variable associated with the host. Log messages are displayed at the bottom of the xpcnfig window.

## Example of the xpcnfig Primary Window

Figure 9-1 is an example of the xpcnfig primary window.

Figure 9-1 xpcnfig Primary Window



## Accessing the xpcnfig Primary Window

You can start the xpcnfig configuration utility and access the xpcnfig primary window from either a PATROL Developer Console or from within a native Unix environment.

If you do not have either a PATROL Developer Console or Unix environment, you must use the pcnfig command-line configuration utility. For information on the pcnfig command-line configuration utility, see Chapter 8, “Using pcnfig to Configure the PATROL Agent.”

### Accessing the xpcnfig Primary Window from a PATROL Developer Console

The recommended method for starting the xpcnfig configuration utility and accessing the xpcnfig primary window is through a PATROL Developer Console.

To start the xpcnfig configuration utility from a PATROL Developer Console and access the xpcnfig primary window, perform the following action:

- » Using MB3, click the icon for the machine running the PATROL Agent and choose **Development => Agent Configuration**.

The system displays the `xpcnfig:hostname` window (the primary window).

### Accessing the xpcnfig Primary Window from Within an Unix System

An alternate method of starting the xpcnfig configuration utility and accessing the xpcnfig primary window is from within a native Unix System environment.

To start the xpcnfig configuration utility and access the xpcnfig primary window from an xterm session, use the xpcnfig command at the command line.

## Before You Begin

Before starting `xpconfig` from the command line, you must run one of the following scripts from the PATROL installation directory:

```
. ./patrolrc.sh (for Korn and Bourne shell) or source
.patrolrc (for C shell). You also need to set the display variable.
```

## Syntax of the Command for the `xpconfig` Configuration Utility

The command for the `xpconfig` configuration utility is `xpconfig`. The format of the `xpconfig` utility is as follows:

```
xpconfig [-host name[, ...]] [-port num]
```

## Options for Starting the `xpconfig` Configuration Utility from the Command Line

Use Table 9-2 to find the `xpconfig` configuration utility option for the task you want to perform.

**Table 9-2 Options for Starting the `xpconfig` from the Command Line**

| Task You Want to Perform                                     | Option to Use                  | Additional Information  |
|--|--------------------------------|---|
| Specify the host names that are in the host list by default. | <code>-host <i>name</i></code> | When the program exits, it saves the current host list to use next time; however, the command line arguments override these settings. |
| Specify the port number to be used.                          | <code>-port <i>num</i></code>  | The default port number for version 3.4 or later is 3181. The default port number for versions 3.3 or earlier was 1987.               |
| Display version information for <code>xpconfig</code>        | <code>-version</code>          |   |

## To Access the `xpconfig` Primary Window from the Command Line

Start the `xpconfig` configuration utility and access the `xpconfig` primary window from an `xterm` session command line by performing the following action:

- » Type the file name `xpconfig` and any host or port options and press **Enter**.

The system displays the `xpconfig:hostname` window (the `xpconfig` primary window).

## Closing the `xpconfig` Primary Window

When you are finished making changes, choose **File => Exit**. If you have not saved the current file, you will get a dialog box asking whether you want to save your changes. If the current file is already saved, the program will exit.

## Cancelling Changes Not Applied

To cancel any changes not applied and to return to the primary window, click **Cancel**.

## Stopping an Operation

While it is not recommended, if you feel you have received enough information to consider an operation complete, you can click **Stop** to halt the operation.

Clicking **Stop** does not abort the operation. The operation is considered complete with whatever information is retrieved when you clicked **Stop**.

## Returning to the `xpconfig` Primary Window

Before returning to the `xpconfig` Primary Window, you must decide whether to save the unapplied changes, abandon these changes, or stop viewing the change file. Choose one of the following methods based on your decision.

## To Save Changes and Return to the xpcnfig Primary Window

» Perform one of the following actions:

- Choose **File => Close**
- Click **OK**

If you added a new variable or changed a default variable, an asterisk (\*) is displayed to the left of the variable name in the primary window.

## To Abandon Any Changes Not Yet Applied and Return to the xpcnfig Primary Window

» Click **Cancel**.

## To Stop Viewing the Change File and Return to the xpcnfig Primary Window

» Choose **File => Exit**.

## Selecting a Change File

How you select a change file depends on whether you know the path of the file.

### To Select a Change File When You Do Not Know the File Path

- Step 1** Click **Filter** to list only the configuration files.
- Step 2** Scroll the **Directories** list and double-click the appropriate directory name.
- Step 3** Scroll the **Files** list and double-click the appropriate file name.
- Step 4** Click **OK**.

### To Select a Change File When You Know the File Path

- Step 1** Type the file path and name in the **Selection** field.
- Step 2** Press **Enter**, or click **OK**.

## When Changes Take Effect

Most of the PATROL Agent's configuration variables take effect immediately after you select **Apply Configuration** in the primary window.

For a detailed listing of the variables that take effect immediately, see the section "When Changes to the Configuration Take Effect" on page 3-7.

For information on reinitializing the PATROL Agent so that the other variables take effect, see the section "Reinitializing the PATROL Agent" on page 9-13.

## Reinitializing the PATROL Agent

---

**Summary:** Use this procedure to reinitialize (restart) the PATROL Agent from the xpcnfig configuration utility.

---

### Before You Begin

Before you begin to reinitialize (restart) the PATROL Agent, make sure that you have completed the following prerequisites:

- You have read and understood the sections “Overview of the xpcnfig Configuration Utility” on page 9-3, and “Overview of xpcnfig Functions” on page 9-5.
- You have started the xpcnfig configuration utility.

### To Reinitialize (Restart) the PATROL Agent

**Step 1** In the primary window of the utility, click the host where you want to reinitialize (restart) the PATROL Agent.

The host is highlighted.

**Step 2** Choose **Options => Reinitialize Agent**.

The Restart Agent dialog box is displayed.

**Figure 9-2 The Restart Agent Dialog Box**



**Step 3** Click **Yes**.

The agent restarts and the system displays a series of messages regarding the restart of the agent.

**Step 4** If you have a PATROL Console running, in the console's main window, reconnect the PATROL Agent by using MB3 to click the icon for the machine running the PATROL Agent and choosing **Update Connection**.

If you are using PATROL Central Operator - Microsoft Windows Edition, use the **Connect to Managed System** command to update the connection.

## Reloading the config.default File and Change File for the PATROL Agent

---

**Summary:** Use this procedure to reload the **config.default** file and the change file for the PATROL Agent from the xpcnfig configuration utility. This procedure is helpful if the **config.default** has been modified.

---

### Before You Begin

Before you begin to reload the **config.default** file and the change file for the PATROL Agent, make sure that you have completed the following prerequisites:

- You have read and understood the sections “Overview of the xpcnfig Configuration Utility” on page 9-3, and “Overview of xpcnfig Functions” on page 9-5.
- You have started the xpcnfig configuration utility.

### To Reload the config.default File and Change File for the PATROL Agent

**Step 1** In the primary window of the utility, click the host where you want to reload the **config.default** file and the change file for the PATROL Agent.

The host is highlighted.

**Step 2** Choose **Options => Reload**.

The Reload Configuration dialog box is displayed.

**Figure 9-3** The Reload Configuration Dialog Box



**Step 3** Click **Yes**.

The agent reloads and shows the changes made to the **config.default** file and the change file. Changes that require the agent to reinitialize are not displayed.

## Stopping the PATROL Agent

---

**Summary:** Use this procedure to stop the PATROL Agent from the xpcnfig configuration utility.

---

### Before You Begin

Before you stop the PATROL Agent, make sure that you have completed the following prerequisites:

- You have read and understood the sections “Overview of the xpcnfig Configuration Utility” on page 9-3, and “Overview of xpcnfig Functions” on page 9-5.
- You have started the xpcnfig configuration utility.

### To Stop the PATROL Agent

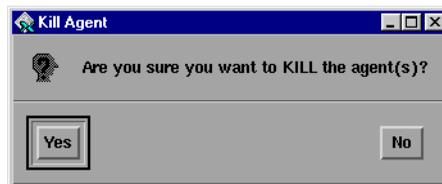
**Step 1** At the primary window of the utility, click the host where you want to stop the PATROL Agent.

The host is highlighted.

**Step 2** Choose **Options => Kill Agent**.

The Kill Agent dialog box is displayed.

**Figure 9-4 The Kill Agent Dialog Box**



**Step 3** Click **Yes**. The agent process is stopped.

# Handling Change Files

Change files are handled from the primary window of the xpcnfig configuration utility. The primary window is displayed when you start the xpcnfig configuration utility.

## Tasks Available for Handling Change Files

You can perform the following tasks for handling change files from the primary window of the xpcnfig configuration utility.

| <b>Task To Perform</b>   | <b>Section to Go to</b>   |
|--|---|
| Opening an existing change file.   | "Opening an Existing Change File" on page 9-19                        |
| Create a new change file from a PATROL Agent's configuration   | "Creating a New Change File" on page 9-21                             |
| Send a complete set of variables to the PATROL Agents to use as the defaults   | "Sending a Complete Set of Variables" on page 9-22                    |
| View a change file   | "Viewing a Change File" on page 9-23                                  |
| Save a change file   | "Saving a Change File" on page 9-24                                   |
| Apply the changes in a change file to a PATROL Agent's configuration   | "Applying the Changes to a PATROL Agent" on page 9-26                 |
| Purge (remove) the existing configuration from a PATROL Agent and create another one from the <b>config.default</b> file and the change file | "Purging the Existing Configuration from a PATROL Agent" on page 9-28 |
| Send a license file to a PATROL Agent  | "Sending a New License File to a PATROL Agent" on page 9-30           |

## Opening an Existing Change File

---

**Summary:** Use this procedure to open a previously saved configuration file containing changes to the default configuration. This procedure is useful if you want to apply specific changes to one or more PATROL Agents.

---

### Before You Begin

Before you begin to open an existing change file, make sure you have completed the following prerequisites:

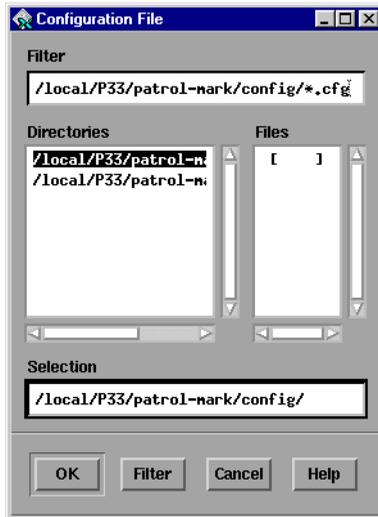
- You have read and understood Chapter 3, “Background About Configuring the PATROL Agent.”
- You have read and understood the sections “Overview of the xpcnfig Configuration Utility” on page 9-3, and “Overview of xpcnfig Functions” on page 9-5.
- You have started the xpcnfig configuration utility.

## To Open an Existing Change File

**Step 1** At the primary window of the utility, choose **File => Open**.

The Configuration File dialog box is displayed. The default location containing a list of files with the cfg extension is \$PATROL\_CACHE/config.

**Figure 9-5 The Configuration File Dialog Box**



**Step 2** Click the configuration file that you want.

**Step 3** Click **OK**.

The primary window is displayed with the selected file in the title bar at the top of the window. All of the variables are displayed in the Variables list.

## Creating a New Change File

---

**Summary:** Use this procedure to create a new change file from an existing agent's configuration. This file contains the changes that have been made to the default configuration and applied to the specified agent's configuration.

---

### Before You Begin

Before you begin to create a new change file from a PATROL Agent's configuration, make sure you have completed the following prerequisites:

- You have read and understood Chapter 3, "Background About Configuring the PATROL Agent."
- You have read and understood the sections "Overview of the xpcnfig Configuration Utility" on page 9-3, and "Overview of xpcnfig Functions" on page 9-5.
- You have started the xpcnfig configuration utility.

### To Create a New Change File

- Step 1** At the primary window, use the up and down arrows to set the port number for the PATROL Agent's host.
- Step 2** Select the host from the list of hosts by scrolling the list and clicking the host name.
- Step 3** Click the **Get Configuration** button at the bottom of the host list.

A series of messages is displayed to show the progress of the operation.

## Sending a Complete Set of Variables

---

**Summary:** Use this procedure to send a complete set of variables to the PATROL Agents to use as the defaults.

---

Before you send a complete set of variables, make sure you have completed the following prerequisites:

- You have read and understood the sections “Overview of the xpcnfig Configuration Utility” on page 9-3, and “Overview of xpcnfig Functions” on page 9-5.
- You have started the xpcnfig configuration utility.

### To Send a Complete Set of Variables

**Step 1** Use the up and down arrows to set the port number for the agent’s host.

**Step 2** From the list of hosts, select the hosts to which you want to send a complete set of variables for the PATROL Agent to use as the defaults. Choose one of the following methods:

- Scroll the list and click the host name of each host you want to configure.
- Choose **Host => Select All** (or use MB3 to choose **Select All** when the cursor is in the host name area) to select all the host names in the list.

**Step 3** Choose **Options => Send Defaults**.

A series of messages is displayed to show the progress of the operation.

## Viewing a Change File

---

**Summary:** Use this procedure to view the contents of a change file. You search for entries while viewing the file.

---

### Before You Begin

Before you begin to view a change file, make sure you have completed the following prerequisites:

- You have read and understood the sections “Overview of the xconfig Configuration Utility” on page 9-3, and “Overview of xconfig Functions” on page 9-5.
- You have started the xconfig configuration utility.

### To View a Change File and Search for Entries

**Step 1** Choose **Options => View**.

The View *view only* dialog box with the contents of the configuration change file is displayed.

The configuration change file contains entries only for the variables marked with an asterisk; the others are default variables with the default value and are already in the **config.default** file.

**Step 2** Begin a search for entries by choosing **Edit => Search**.

**Step 3** Type the text to search for (the search function is case-sensitive), and click **OK**.

**Step 4** Exit the View *view only* dialog box by choosing **File => Exit**.

The system displays the primary window.

## Saving a Change File

---

**Summary:** Use this procedure to save a change file. You would perform this procedure to apply these changes to one or more agents some time in the future.

---

### Before You Begin

Before you begin to save a change file, make sure you have completed the following prerequisites:

- You have read and understood Chapter 3, “Background About Configuring the PATROL Agent.”
- You have read and understood the sections “Overview of the xpcnfig Configuration Utility” on page 9-3, and “Overview of xpcnfig Functions” on page 9-5.
- You have started the xpcnfig configuration utility.

The change file you save contains entries only for the variables marked with an asterisk; the others are default variables with the default value and are already in the **config.default** file.

### To Save a Change File

**Step 1** Use one of the following methods:

- If you want to save the file to the same name, choose **File => Save**.

The change file is saved.

- If you want to save the file under a different file name, choose **File => Save As**.

The Configuration File dialog box is displayed. (See Figure 9-5 on page 9-20.)

**Step 2** Click the change file or type the name of the change file.

**Step 3** Click **OK**.

The change file is saved.

## Applying the Changes to a PATROL Agent

---

**Summary:** Once all the changes have been added to your configuration file, you can apply them to one or more PATROL Agents by using the following procedure. The configuration utility sends the new or changed variables to the PATROL Agent. The PATROL Agent updates the change database (`$PATROL_HOME/config/config_hostname-port`).

---

### Before You Begin

Before you begin to apply the change in a change file to a PATROL Agent's configuration, make sure you have completed the following prerequisites:

- You have read and understood Chapter 3, "Background About Configuring the PATROL Agent."
- You have read and understood the sections "Overview of the xpcfg Configuration Utility" on page 9-3, and "Overview of xpcfg Functions" on page 9-5.
- You have started the xpcfg configuration utility.
- You have made the changes you want to apply.

---

### Warning

---

Take care to only apply configuration changes that you intend to change. Pay special attention to any asterisks that are displayed next to variables listed in the Variables Box of the Primary Window. An asterisk means that the variable has changed. Review all modified variables before clicking the Apply Configuration button. If you do not want to apply the change associated with a variable that is marked by an asterisk, you must delete the variable from the Variables Box before clicking the Apply Configuration button.

---

## To Apply the Changes

**Step 1** Use the up and down arrows to set the port number for the agent's host.

**Step 2** From the list of hosts, select the hosts to be configured with this configuration. Use one of the following methods:

- Scroll the list and click the host name of each host you want to configure.

The selected hosts are highlighted.

- Choose **Host => Select All** (or use MB3 to choose **Select All** when the cursor is in the host name area) to select all the host names in the list.

The selected hosts are highlighted.

**Step 3** Click **Apply Configuration** at the bottom of the window.

The changes are applied, and the system displays a series of messages that show the progress of the operation.

---

### Note

---

Modifying some variables require reinitializing the agent. Verify whether any variable that you have modified has this requirement. Restart the agent if this is true for one or more variables.

---

## Purging the Existing Configuration from a PATROL Agent

---

**Summary:** You can purge, or remove, the existing configuration from a PATROL Agent by using the following procedure. Purge is used to restore PATROL Agent to its default as specified in **config.default**. The information displayed in the Variable listing (xpconfig Primary Window) is not changed.

---

### Before You Begin

Before you begin to purge (remove) the existing configuration from a PATROL Agent, make sure you have completed the following prerequisites:

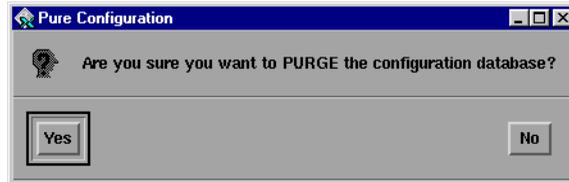
- You have read and understood Chapter 3, “Background About Configuring the PATROL Agent.”
- You have read and understood the sections “Overview of the xpconfig Configuration Utility” on page 9-3, and “Overview of xpconfig Functions” on page 9-5.
- You have started the xpconfig configuration utility.

### To Purge the Existing Configuration from a PATROL Agent

- Step 1** Use the up and down arrows to set the port number for the PATROL Agent’s host.
- Step 2** From the list of hosts, select the hosts to be purged. Choose one of the following methods:
- Scroll the list and click the host name of each host you want to configure.
  - Choose **Host => Select All** (or use MB3 to choose **Select All** when the cursor is in the host name area) to select all host names in the list.
- Step 3** Choose **Options => Purge**.

The Purge Configuration dialog box is displayed.

**Figure 9-6 The Purge Configuration Dialog Box**



**Step 4** Click **Yes**.

The configuration file is purged and the system displays a series of messages that show the progress of the operation.

## Sending a New License File to a PATROL Agent

---

**Summary:** Use this procedure to send a new license file to a PATROL Agent on selected hosts.

---

### Before You Begin

Before you begin to send a new license file to a PATROL Agent, make sure you have completed the following prerequisites:

- You have read and understood Chapter 3, “Background About Configuring the PATROL Agent.”
- You have read and understood the sections “Overview of the xpcnfig Configuration Utility” on page 9-3, and “Overview of xpcnfig Functions” on page 9-5.
- You have started the xpcnfig configuration utility.
- You know the name of the license file you want to send.

### To Send a New License File to a PATROL Agent

**Step 1** Use the up and down arrows to set the port number for the PATROL Agent’s host.

**Step 2** From the list of hosts, select the hosts to which you want to send the license. Choose one of the following methods:

- Scroll the list and click the host name of each host you want to configure.
- Choose **Host => Select All** (or use MB3 to choose **Select All** when the cursor is in the host name area) to select all host names in the list.

**Step 3** Choose **Options => License**.

The File Selection dialog box is displayed.

**Figure 9-7 The File Selection Dialog Box**



**Step 4** Select the license file you want to send to the PATROL Agent.

**Step 5** Click **OK**.

The Send License File dialog box is displayed.

**Step 6** Click **Yes**.

The license file is sent to the selected hosts and the system displays a series of messages showing the progress of the operation.

# Handling Variables

Variables are handled from the primary window and the Edit Variable dialog box of the xpconfig configuration utility. The primary window is displayed when you start the xpconfig configuration utility. The Edit Variable dialog box is displayed when you select a variable to edit.

## Handling Variables from the Primary Window

The following table lists tasks that you can perform to handle variables from the primary window of the xpconfig configuration utility.

**Table 9-3 Tasks to Handle Variables from the Primary Window**

| <b>Task to Perform</b>                 | <b>Section to Go to</b>                                  |
|--|--|
| Add a host to the host list            | "Adding a Host to the Host List" on page 9-33            |
| Add new variables to the change file   | "Adding New Variables to the Change File" on page 9-36   |
| Delete a variable from the change file | "Deleting a Variable from the Change File" on page 9-38  |
| Reset a variable to its default value  | "Resetting a Variable to Its Default Value" on page 9-40 |

## Adding a Host to the Host List

---

*Summary:* Use this procedure to add a host that is not on the host list.

---

### Before You Begin

Before you begin to add a host to the host list, make sure you have completed the following prerequisites:

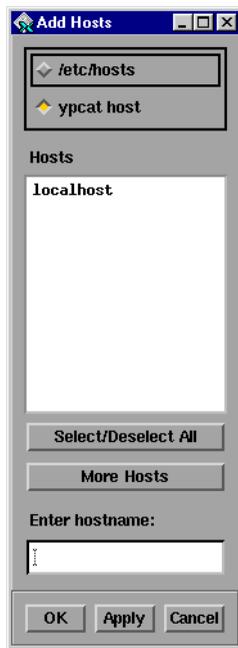
- You have read and understood Chapter 3, “Background About Configuring the PATROL Agent.”
- You have read and understood the section “Overview of xpcnfig Functions” on page 9-5.
- You have started the xpcnfig configuration utility, and the primary window is displayed.

### To Add a Host to the Host List

**Step 1** With the mouse pointer over the host list box, use MB3 to choose **Add**, or choose **Host => Add**.

The Add Hosts dialog box is displayed.

**Figure 9-8 The Add Hosts Dialog Box**



**Step 2** Click **/etc/host** or **ypcat host**, depending on which list of hosts you want to select from.

The system displays the appropriate host list.

**Step 3** Perform one of the following actions:

- Select a host from the list.
- Type a host name in the **Enter hostname** field.

**Step 4** Perform one of the following actions:

- To add the name to the host list and remain in the Add Hosts dialog box to select another name, click **Apply**.

The host is added to the host list.

- To add the name to the host list and close the Add Hosts dialog box, click **OK**.

The host is added to the host list and the system displays the primary window.

## Adding New Variables to the Change File

---

**Summary:** Use the following procedure to add a new variable to the change file.

---

### Before You Begin

Before you begin to add a new variable to the change file, make sure you have completed the following prerequisites:

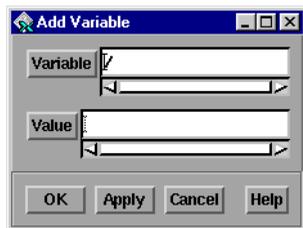
- You have read and understood Chapter 3, “Background About Configuring the PATROL Agent.”
- You have read and understood the section “Overview of xpcnfig Functions” on page 9-5.
- You have started the xpcnfig configuration utility, and the primary window is displayed.
- You know the variable you want to add.

### To Add a New Variable to the Change File

**Step 1** Choose **Edit => Add** from the menu bar.

The Add Variable dialog box is displayed.

**Figure 9-9 The Add Variable Dialog Box**



**Step 2** In the **Variable** field, type the variable name in the form */path/variable\_name*. If the variable does not exist, it will be created.

**Step 3** In the **Value** field, type the variable's value. If the value is a list, separate the items by commas since the append, merge, and replace operations require commas to distinguish the values.

**Step 4** Perform one of the following actions:

- To add the new variable to the change file and leave the Add Variable dialog box open to add another variable, click **Apply**.

The variable is added to the change file.

- To add the new variable, close the dialog box, return to the primary window, and click **OK**.

The variable is added to the change file and the system displays the primary window.

## Deleting a Variable from the Change File

---

**Summary:** Use the following procedure to delete a variable from the change file.

---

### Before You Begin

Before you begin to delete a variable from the change file, make sure you have completed the following prerequisites:

- You have read and understood Chapter 3, “Background About Configuring the PATROL Agent.”
- You have read and understood the section “Overview of xconfig Functions” on page 9-5.
- You have started the xconfig configuration utility, and the primary window is displayed.
- You know the variable you want to delete.

### To Delete a Variable from the Change File

**Step 1** From the xconfig Primary Window, double-click the desired variable listed in the Variable Box.

The variable (to be deleted) is displayed in the Edit Variable dialog box.

**Step 2** Delete everything in the Change Field.

**Step 3** Click the Delete Radio Button.

**Step 4** Click the Apply Button.

**Step 5** Click the OK Button.

The Edit Variable dialog box closes to reveal the xconfig Primary Window.

- Step 6** Select **Options => View** to verify that the message to be sent to the agent includes a delete action for the variable.
- Step 7** Click the **Apply Configuration Button**.
- Step 8** Click the **Get Configuration Button** to verify that the variable is deleted.

## Resetting a Variable to Its Default Value

---

**Summary:** Use this procedure to reset a variable to its default value.

---

### Before You Begin

Before you begin to reset a variable to its default value, make sure you have completed the following prerequisites:

- You have read and understood Chapter 3, “Background About Configuring the PATROL Agent.”
- You have read and understood the section “Overview of xconfig Functions” on page 9-5.
- You have displayed the xconfig Primary Window.
- You know the variables to be reset to the default value.

### To Reset a Variable to Its Default Value

**Step 1** From the xconfig Primary Window, click the desired variable listed in the Variable Box to highlight it.

The variable (to be reset) is displayed in the Edit Variable dialog box.

**Step 2** Click the Edit Button.

**Step 3** Click the Reset to Default Button.

**Step 4** Click the Apply Configuration Button.

**Step 5** Click the Get Configuration Button to verify the procedure.

## Handling Variables from the Edit Variable Dialog Box

Table 9-4 presents tasks that you can perform to handle variables from the Edit Variable dialog box of the xconfig configuration utility.

**Table 9-4 Handling Variables from the Edit Variable Dialog Box**

| <b>Task to Perform</b>                                  | <b>Section to go to</b>   |
|---|---|
| Add a Change Entry in the Edit Variable dialog box      | "Inserting a New Change Entry in the Edit Variable Dialog Box" on page 9-44 |
| Modify a Change Entry in the Edit Variable dialog box   | "Modifying a Change Entry in the Edit Variable Dialog Box" on page 9-46     |
| Delete a Change Entry from the Edit Variable dialog box | "Deleting a Change Entry from the Edit Variable Dialog Box" on page 9-48    |
| Modify the Default Account Variable in the Change File  | "Modifying the Default Account Variable in the Change File" on page 9-49    |

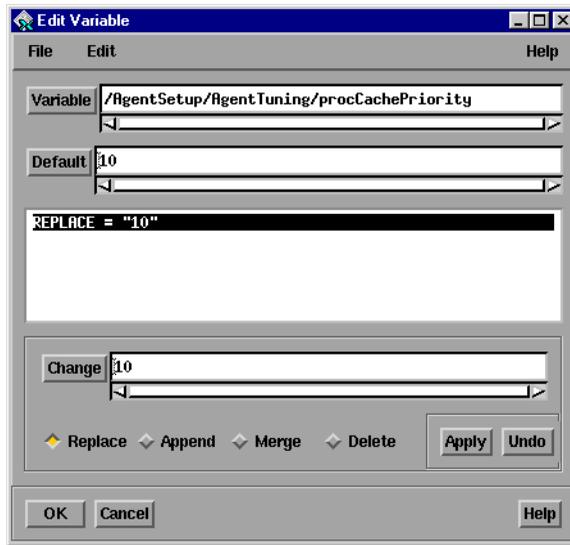
## Modifying Variables Using the Edit Variable Dialog Box

To modify one of the existing variables in the change file, first open the Edit Variable dialog box by using one of the following methods:

- double-click the variable
- click the variable, then use MB3 to choose **Modify**
- click the variable, then choose **Edit => Modify**

The Edit Variable dialog box is displayed.

**Figure 9-10 The Edit Variable Dialog Box**



The variable name and value (or default value, if it is a default variable) are shown in the two fields at the top of the dialog box. You cannot edit these fields. Any change entries for the variable are listed in the middle of the Edit Variable dialog box.

## Methods for Modifying a Long Text Entry in the Edit Variable Dialog Box

If the value you want to enter is particularly long or in a separate file, you can either type it or import it:

- To type a value, click **Change** to see the Change *applied directly* dialog box. You can type long values into this box and see more of the text as you type.
- To import text, choose **File => Load**. The Text File dialog box is displayed.

You can restore the **Change** field to the value of the selected change entry by clicking **Undo**.

1. Select a configuration file.
2. When you have edited this text, choose **File => Exit**. You will get a dialog box asking whether you want to save your changes.
3. Click **Apply** to add your modification to the change entry.

## Inserting a New Change Entry in the Edit Variable Dialog Box

---

**Summary:** Use this procedure to insert a new change entry for a variable that already has a change entry in the Edit Variable dialog box.

---

### Before You Begin

Before you begin to insert a new change entry in the Edit Variable dialog box, make sure you have completed the following prerequisites:

- You have read and understood Chapter 3, “Background About Configuring the PATROL Agent.”
- You have read and understood the sections “Overview of xpcnfig Functions” on page 9-5, and “Handling Variables” on page 9-32.
- You have started the xpcnfig configuration utility, opened a change file, and accessed the Edit Variable dialog box by clicking the variable for which you want to insert the change entry.

You can restore the **Change** field to the value of the selected change entry by clicking **Undo**.

### To Insert a Change Entry in the Edit Variable Dialog Box

**Step 1** Click an entry that is before or after where you want the new entry to go.

**Step 2** Use MB3 to choose **Insert Before**, **Insert After**, **Edit => Insert Before**, or **Edit => Insert After**, depending on where you want the new entry to go.

The system displays a `REPLACE = " "` entry in the designated place.

**Step 3** Modify the new entry as described in “Modifying a Change Entry in the Edit Variable Dialog Box” on page 9-46.

**Step 4** Perform one of the following actions:

- To apply the change and close the dialog, click **OK**.

The entry is changed and the system displays the primary window.

- To apply the change and remain in this dialog, click **Apply**.

The entry is changed.

- To abandon any changes not applied yet, click **Cancel**.

The entry is canceled and the system displays the primary window.

## Modifying a Change Entry in the Edit Variable Dialog Box

---

**Summary:** Use this procedure to modify a change entry that is currently in the Edit Variable dialog box.

---

### Before You Begin

Before you begin to modify a change entry in the Edit Variable dialog box, make sure you have completed the following prerequisites:

- You have read and understood Chapter 3, “Background About Configuring the PATROL Agent.”
- You have read and understood the sections “Overview of xpcnfig Functions” on page 9-5, and “Handling Variables” on page 9-32.
- You have started the xpcnfig configuration utility, opened a change file, and accessed the Edit Variable dialog box by clicking the variable for which you want to modify the change entry.

You can restore the **Change** field to the value of the selected change entry by clicking **Undo**.

### To Modify a Change Entry in the Edit Variable Dialog Box

**Step 1** Click the entry you want to modify.

The entry is highlighted.

**Step 2** Enter the new value in the **Change** field. If there is more than one value, separate the values with commas.

**Step 3** Perform one of the following actions:

- To apply the change and close the dialog, click **OK**.

The entry is changed and the system displays the primary window.

- To apply the change and remain in this dialog, click **Apply**.

The entry is changed.

- To abandon any changes not applied yet, click **Cancel**.

The entry is canceled, and the system displays the primary window.

## Deleting a Change Entry from the Edit Variable Dialog Box

---

**Summary:** If you do not want a specified change in the Edit Variable dialog box to take effect, you can delete the entry.

If the Edit Variable dialog box has other entries that you do not delete, deleting the one change entry does not delete the variable. However, if you delete all change entries for a variable, the variable is deleted from the change file.

---

### Before You Begin

Before you begin to delete a change entry from the Edit Variable dialog box, make sure you have completed the following prerequisites:

- You have read and understood Chapter 3, “Background About Configuring the PATROL Agent.”
- You have read and understood the sections “Overview of xpcnfig Functions” on page 9-5, and “Handling Variables” on page 9-32.
- You have started the xpcnfig configuration utility, opened a change file, and accessed the Edit Variable dialog box by clicking the variable for which you want to delete the change entry.

### To Delete a Change Entry in the Edit Variable Dialog Box

**Step 1** Click the entry you want to delete.

The entry is highlighted.

**Step 2** Use MB3 to choose **Delete**, or choose **Edit => Delete**.

## Modifying the Default Account Variable in the Change File

---

**Summary:** Use the following procedure to modify the variable for the user account that the PATROL Agent uses if an account is not explicitly specified.

---

### Before You Begin

Before you begin to modify the default account variable, make sure you have completed the following prerequisites:

- You have read and understood Chapter 3, “Background About Configuring the PATROL Agent.”
- You have read and understood the section “Overview of xpcnfig Functions” on page 9-5.
- You have started the xpcnfig configuration utility.
- You know the change you want to make for the default user account.

When the variable you select to change is the Default Account variable, the Edit Variable dialog box changes.

- The Set Default Account field initially shows the current default account user name.
- The Modify and Append buttons are unavailable because the default account variable cannot take a list as a value.

### To Modify the Default Account Variable in the Change File

**Step 1** Perform one of the following actions:

- To delete the current name, click **Delete** and then click **Apply** or **OK**.

The current name is deleted.

- To change the current name, click **Set Default Account**.

The Set Default Account dialog box is displayed.

**Step 2** Type the account name in the User Name field.

**Step 3** Type the password in the Password field. The password is not displayed as you type it.

**Step 4** Click **OK**.

The system displays the new account name followed by the encrypted password in the Set Default Account field.

**Step 5** Perform one of the following actions:

- To apply the change and close the dialog box, click **OK**.

The name is changed and the system displays the primary window.

- To apply the change and remain in this dialog box, click **Apply**.

The name is changed.

- To abandon any changes not applied yet, click **Cancel**.

The change is canceled, and the system displays the primary window.

---

# Using wpconfig (Windows) to Configure the PATROL Agent

This chapter describes how you can use the wpconfig utility to configure the PATROL Agent. This chapter discusses the following topics and tasks:

|  |       |
|--|-------|
| Overview of the wpconfig Configuration Utility . . . . .             | 10-2  |
| Prerequisites for Using the wpconfig Configuration Utility . . . . . | 10-3  |
| Overview of wpconfig Functions . . . . .                             | 10-3  |
| Working with the wpconfig Window . . . . .                           | 10-6  |
| Accessing the wpconfig Window. . . . .                               | 10-7  |
| Closing the wpconfig Window. . . . .                                 | 10-8  |
| Creating a New Configuration . . . . .                               | 10-8  |
| Tasks Available for Using the wpconfig Utility . . . . .             | 10-8  |
| Retrieving a Configuration from a PATROL Agent. . . . .              | 10-8  |
| Getting a Configuration from a PATROL Agent . . . . .                | 10-9  |
| Adding a Variable to a PATROL Agent Configuration . . . . .          | 10-11 |
| Adding a New Variable . . . . .                                      | 10-12 |
| Modifying a Variable in a Configuration . . . . .                    | 10-14 |
| Modifying an Existing Variable. . . . .                              | 10-15 |
| Modifying the Default Account Variable. . . . .                      | 10-18 |
| Deleting a Variable from a Configuration . . . . .                   | 10-20 |
| Deleting an Existing Variable . . . . .                              | 10-21 |
| Viewing Changes in a Configuration . . . . .                         | 10-22 |
| Applying a Configuration to a PATROL Agent. . . . .                  | 10-25 |
| Applying a Configuration to an Agent. . . . .                        | 10-26 |

# Overview of the wpconfig Configuration Utility

The wpconfig utility provides the following functionality for the configuration of a PATROL Agent:

- get a configuration
- add a new variable
- modify an existing variable
- delete an existing variable
- view the changes you have made
- apply the configuration

The wpconfig utility provides the following functionality for the PATROL Agent:

- reload
- purge
- reinitialize
- stop (kill)

# Prerequisites for Using the wpconfig Configuration Utility

Before you configure the PATROL Agent with the wpconfig configuration utility, make sure you have completed these prerequisites:

- You have reviewed and understood Chapter 3, “Background About Configuring the PATROL Agent.”
- You know the change files you want to use.
- You have started the PATROL Agent. (See Chapter 2, “Starting and Stopping the PATROL Agent.”)

If you want to change values for variables, make sure you have completed these prerequisites:

- You have write permission on the change file.
- You know the variables that you need to change in the change files.

## Overview of wpconfig Functions

Table 10-1 provides an overview of the menus and menu commands included in wpconfig.

**Table 10-1 Menus and Menu Commands for the wpconfig Utility (Part 1 of 2)**

| <b>Menu</b> | <b>Command</b>   | <b>Description</b>  |
|-------------|------------------|---|
| File        | New              | Creates a new PATROL Agent configuration by loading the <code>config.default</code> file.                                     |
|             | Open             | Opens an existing PATROL Agent configuration by loading the <code>config.default</code> file and the change file you specify. |
|             | Save             | Saves the change file for the PATROL Agent.   |
|             | Save As          | Saves the change file for the PATROL Agent to a name you specify.   |
|             | Exit             | Closes the wpconfig window.   |
| Edit        | Undo             | Reverses the last action.   |
|             | Cut              | Deletes the selected text and moves it to the clipboard.  |
|             | Copy             | Copies the selected text to the clipboard.  |
|             | Paste            | Inserts the text from the clipboard.  |
|             | Select All       | Selects all variables in the configuration.   |
|             | Add Variable     | Displays the Add Variable dialog box to insert a new variable into the configuration.   |
|             | Modify Variable  | Displays the Modify Variable dialog box to modify an existing variable in the configuration.                                  |
|             | Delete Variable  | Displays the Delete Variable dialog box to delete an existing variable from the configuration.                                |
|             | Reset to Default | Resets the selected variable to its default value.  |
| View        | Toolbar          | Displays and hides the wpconfig Toolbar.  |
|             | Status Bar       | Displays and hides the wpconfig Status Bar.   |

**Table 10-1 Menus and Menu Commands for the wpconfig Utility (Part 2 of 2)**

| <b>Menu</b> | <b>Command</b>            | <b>Description</b>  |
|-------------|---------------------------|---|
| Tools       | Get Configuration         | Displays the Get Configuration dialog box to open an existing PATROL Agent configuration.   |
|             | Apply Configuration       | Displays the Apply Configuration dialog box to send the currently opened configuration to a PATROL Agent.   |
|             | View Changes              | Displays the View Changes window to show the changes made to the configuration.   |
|             | Reload Agent              | Reloads the <code>config.default</code> file and the change file for a PATROL Agent.  |
|             | Purge Agent               | Removes the existing configuration from a PATROL Agent and builds a new configuration from the <code>config.default</code> file and the change file for the PATROL Agent.   |
|             | Reinitialize Agent        | Displays the Reinitialize Agent dialog box to restart the PATROL Agent.   |
|             | Kill Agent                | Displays the Kill Agent dialog box to stop the PATROL Agent.  |
|             | Update License            | Displays the Open dialog box to open a license file to send to a PATROL Agent.  |
| Options     | Include Default Variables | Specifies whether to include the default variables and their values when saving and applying configuration changes. The default value for this option is <code>off</code> so that default variables are not included. |
|             | Purge Before Applying     | Specifies whether or not to purge a remote agent configuration before applying a new configuration. The default value for this option is <code>off</code> so that the remote agent is not purged.                     |
| Help        | About wpconfig            | Displays the About box containing the version number of wpconfig and other information about PATROL.  |

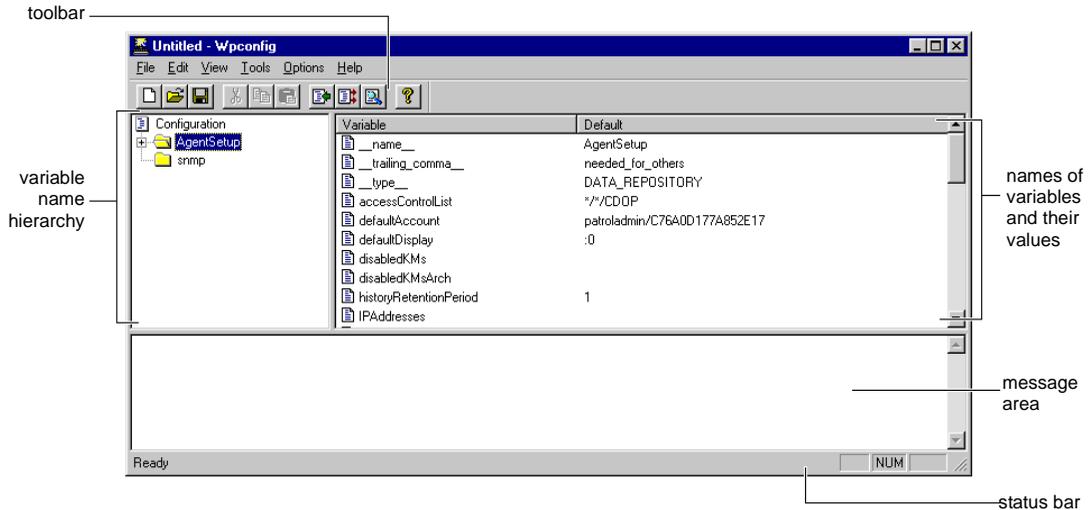
# Working with the wpconfig Window

The wpconfig window displays the current configuration in an interface that is similar to Windows Explorer. You can navigate through the variable name hierarchy in the tree on the left and view the names of the variables in the current section in the list on the right. Log messages are displayed at the bottom of the wpconfig window.

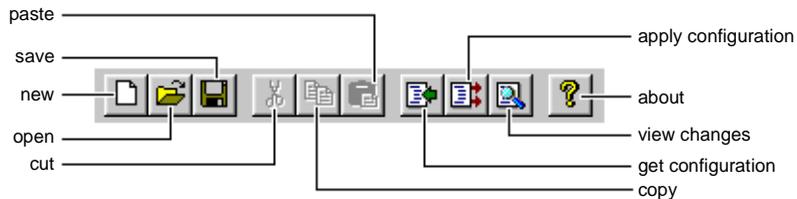
## Example of the wpconfig Window

Figure 10-1 is an example of the wpconfig window.

**Figure 10-1 wpconfig Window**



**Figure 10-2 wpconfig Toolbar**



## Accessing the wpconfig Window

You can start the wpconfig configuration utility and access the wpconfig primary window using one of the following:

- Windows Start Menu Run utility
- Windows Explorer
- the DOS prompt

If you do not have either a PATROL Developer Console or Windows environment, you must use the pconfig command-line configuration utility. For information on the pconfig command-line configuration utility, see Chapter 8, “Using pconfig to Configure the PATROL Agent.”

### Access Control List

The agent access control list (ACL) defines the wpconfig connection mode as configuration (C).

### To Access the wpconfig Window

**Step 1** From the Windows Start Button, click **Start => Run**.

The Run dialog box is displayed.

**Step 2** Type `%PATROL_HOME%\bin\wpconfig.exe` and click **OK**.

The wpconfig window is displayed.

---

**Note**

---

In Step 2, the `-v` option can be used to display the version number of wpconfig.

---

## Closing the wpconfig Window

You can close the wpconfig window by choosing **File => Exit**.

## Creating a New Configuration

When wpconfig is started, a new configuration is created using the variables and values specified in the local **config.default** file as default variables. If a change file was specified, either as a command-line argument or by double-clicking its icon in the Open dialog box, then the specified changes are loaded also.

## Tasks Available for Using the wpconfig Utility

You can perform these tasks using the wpconfig configuration utility:

- retrieve a configuration from a PATROL Agent
- add a variable to a PATROL Agent configuration
- modify a variable in a configuration
- delete a variable from a configuration
- view changes in a configuration
- apply a configuration to a PATROL Agent

## Retrieving a Configuration from a PATROL Agent

You can retrieve a configuration from a remote PATROL Agent. The default variables for the PATROL Agent are retrieved as well as any changes made to the PATROL Agent's current configuration.

## Getting a Configuration from a PATROL Agent

---

**Summary:** Use this procedure to get a configuration from a remote PATROL Agent.

---

### Before You Begin

Before you begin to get a configuration from the PATROL Agent, make sure that you have completed the following prerequisites:

- You have read and understood the sections “Overview of the wpconfig Configuration Utility” on page 10-2, and “Overview of wpconfig Functions” on page 10-3.
- You have started the wpconfig configuration utility.

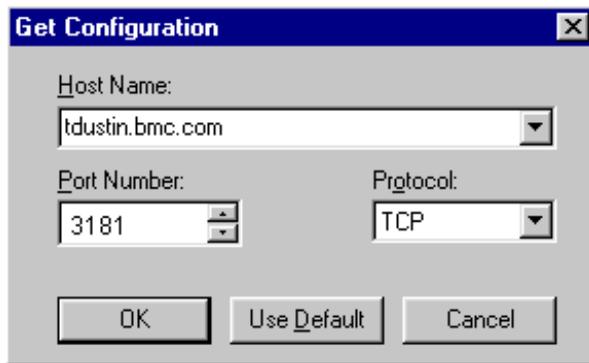
## To Get a Remote Agent's Configuration

**Step 1** Perform one of the following actions:

- Choose **Tools => Get Configuration**.
- Click the Get Configuration toolbar button.

The Get Configuration dialog box is displayed. The dialog box is initialized using the values from the last agent communication.

**Figure 10-3 The Get Configuration Dialog Box**



**Step 2** Enter or select (from the drop down list) the host name of the remote agent as well as the port and protocol to use when establishing the connection.

**Step 3** Click **OK**.

The default variables for the PATROL Agent and any changes that have been applied are displayed.

## Adding a Variable to a PATROL Agent Configuration

After you open or get a configuration and it is displayed in the main window, you can edit the configuration by adding new variables. The new variables are displayed with the existing variables in the list of variables and their values. Any leading or trailing white space is trimmed from the variable name and value string before being added to the configuration. In addition, leading and trailing double-quotes are trimmed from the value setting.

## Adding a New Variable

---

**Summary:** Use this procedure to add a new variable to the agent's configuration.

---

### Before You Begin

Before you begin to add a new variable, make sure that you have completed the following prerequisites:

- You have read and understood the sections “Overview of the wpcnfig Configuration Utility” on page 10-2, and “Overview of wpcnfig Functions” on page 10-3.
- You have started the wpcnfig configuration utility.

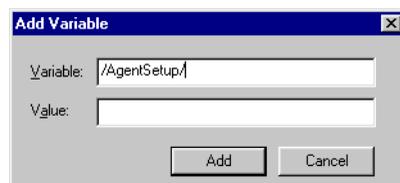
### To Add a New Variable to the Configuration

**Step 1** Perform one of the following actions:

- Choose **Edit => Add Variable**.
- Press the **Insert** key.

The Add Variable dialog box is displayed. The variable name is initialized using the current section in the variable name hierarchy.

**Figure 10-4 The Add Variable Dialog Box**



**Step 2** Type the name of the variable and its value.

**Step 3** Click **OK**.

The variable is added to the list of variables and their values. If a variable with the same name already exists in the configuration, a prompt is displayed for a different variable name.

## Modifying a Variable in a Configuration

After you open or get a configuration and it is displayed in the main window, you can edit the configuration by modifying the variables. You modify variables by editing the change entries for the variable in the Change Entry dialog box, which can be accessed from the Modify Variable dialog box.

When you modify a variable, any leading or trailing white space is trimmed from the variable name and value string before being added to the configuration. In addition, leading and trailing double-quotes are trimmed from the value string.

## Modifying an Existing Variable

---

**Summary:** Use this procedure to modify an existing variable in a configuration.

---

### Before You Begin

Before you begin to modify an existing variable, make sure that you have completed the following prerequisites:

- You have read and understood the sections “Overview of the wpconfig Configuration Utility” on page 10-2, and “Overview of wpconfig Functions” on page 10-3.
- You have started the wpconfig configuration utility.

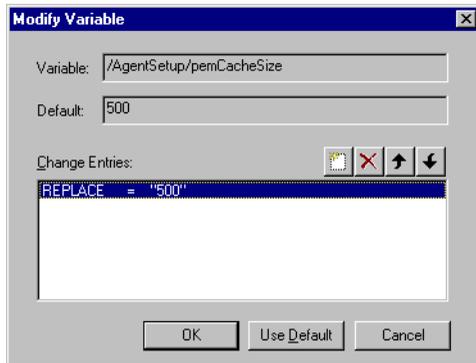
### To Modify a Variable in the Configuration

**Step 1** Perform one of the following actions:

- Click the variable in the list of variables and their values and choose **Edit => Modify Variable**.
- Double-click the variable.

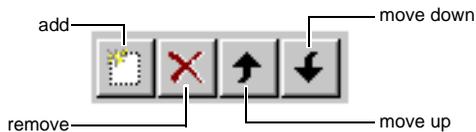
The Modify Variable dialog box is displayed. The dialog box is initialized using the values from the variable you selected. The variable name and any existing default value are displayed as read-only with the change entries for the variable in a scrollable list box below them.

**Figure 10-5 The Modify Variable Dialog Box**



The toolbar in the Modify Variable dialog box, as shown in Figure 10-6, is used to initiate editing operations for the change entries in the list box. The buttons allow the user to add, delete, move up, and move down the entries in the list.

**Figure 10-6 Toolbar in the Modify Variable Dialog Box**

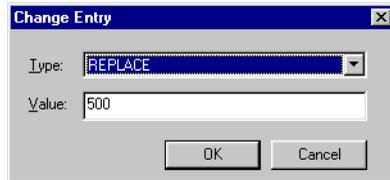


**Step 2** Perform one of the following actions:

- Double-click the change entry.
- Click the appropriate toolbar button.

The Change Entry dialog box is displayed.

**Figure 10-7 The Change Entry Dialog Box**



**Step 3** Specify the type of change entry you want and the value for the change.

**Step 4** Click **OK** in the Change Entry dialog box.

**Step 5** Click **OK** in the Modify Variable dialog box.

The change entry is added to the variable and a red icon is displayed next to the name of the variable that has been changed.

## Modifying the Default Account Variable

---

**Summary:** After you open or get a configuration and it is displayed in the main window, you can modify the default account variable. The variable name for the default account variable is `AgentSetup/defaultAccount`.

---

### Before You Begin

Before you begin to modify the default account variable, make sure that you have completed the following prerequisites:

- You have read and understood the sections “Overview of the wpconfig Configuration Utility” on page 10-2, and “Overview of wpconfig Functions” on page 10-3.
- You have started the wpconfig configuration utility.

---

#### Note

---

The pconfig and xpconfig utilities can also be used to modify this account. See “Using xpconfig (Unix) to Configure the PATROL Agent” on page 9-1 and “Using pconfig to Configure the PATROL Agent” on page 8-1.

---

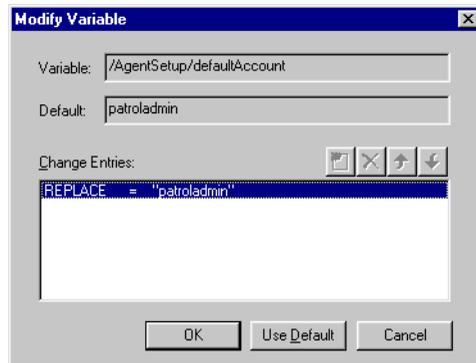
### To Modify the Default Account Variable

**Step 1** Perform one of the following actions:

- Click the default account variable in the list of variables and their values and choose **Edit => Modify Variable**.
- Double-click the default account variable.

The Modify Variable dialog box is displayed.

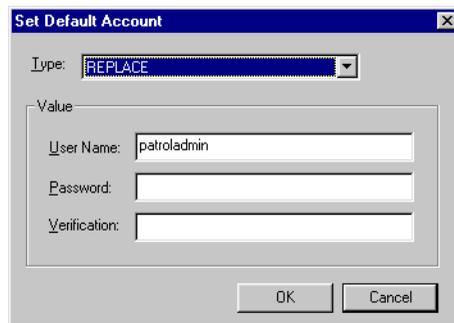
**Figure 10-8 The Modify Variable Dialog Box**



**Step 2** Double-click the existing change entry.

The Set Default Account dialog box is displayed.

**Figure 10-9 The Set Default Account Dialog Box**



**Step 3** Specify the user name and password for the default user account. You must type the password twice for verification.

**Step 4** Click **OK** in the Set Default Account dialog box.

**Step 5** Click **OK** in the Modify Variable dialog box.

The change entry is added to the variable and a red icon is displayed next to the name of the variable that has been changed.

## Deleting a Variable from a Configuration

After you open or get a configuration and it is displayed in the main window, you can edit the configuration by deleting variables. The deleted variables and their values are removed from the list of variables.

Deleting the variable means that it is not included when applying a configuration to a remote agent or when saving to disk. Deleting does not remove the variable from a remote agent's configuration. To delete a variable from a remote agent's configuration, modify the variable to include the "DELETE" change command in the variable's change entry list.

---

### Note

---

You cannot delete default variables from the configuration. If you attempt to delete a default variable, a warning message says that the variable cannot be deleted, and the variable is left unchanged.

---

## Deleting an Existing Variable

---

**Summary:** Use this procedure to delete an existing variable from a configuration.

---

### Before You Begin

Before you begin to delete an existing variable from the configuration, make sure that you have completed the following prerequisites:

- You have read and understood the sections “Overview of the wpconfig Configuration Utility” on page 10-2, and “Overview of wpconfig Functions” on page 10-3.
- You have started the wpconfig configuration utility.

### To Delete Variables from the Configuration

**Step 1** Click the variables in the configuration variable list.

**Step 2** Perform one of the following actions:

- Choose **Edit => Delete Variable** menu item.
- Press the **Delete** key.

The variable and its value is removed from the list of variables.

## Viewing Changes in a Configuration

After you have modified the configuration displayed in the main window, you can view the resulting change file to be used when the configuration is applied to a remove agent or saved to a file.

## Viewing Configuration Changes

---

**Summary:** Use this procedure to view the changes made in a configuration.

---

### Before You Begin

Before you begin viewing changes, complete the following prerequisites:

- You have read and understood the sections “Overview of the wpconfig Configuration Utility” on page 10-2, and “Overview of wpconfig Functions” on page 10-3.
- You have started the wpconfig configuration utility.

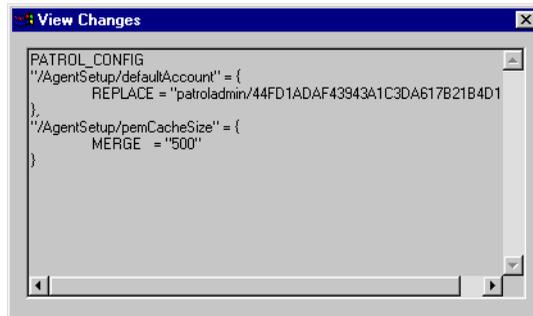
### To View the Configuration Changes

» Perform one of the following actions:

- Choose **Tools => View Changes**.
- Click the View Changes toolbar button.

The View Changes dialog box is displayed.

**Figure 10-10 The View Changes Dialog Box**



## Applying a Configuration to a PATROL Agent

After you have modified the configuration displayed in the main window, you can apply the configuration to a remote PATROL Agent to update the PATROL Agent's configuration.

Some of the variables require the PATROL Agent to be reinitialized before they take effect. For information on when changes to configuration variables take effect, see "When Changes to the Configuration Take Effect" on page 3-7.

## Applying a Configuration to an Agent

---

**Summary:** Use this procedure to apply a configuration to an agent.

---

### Before You Begin

Before you begin to apply a configuration to an PATROL Agent, make sure that you have completed the following prerequisites:

- You have read and understood the sections “Overview of the wpconfig Configuration Utility” on page 10-2, and “Overview of wpconfig Functions” on page 10-3.
- You have started the wpconfig configuration utility.

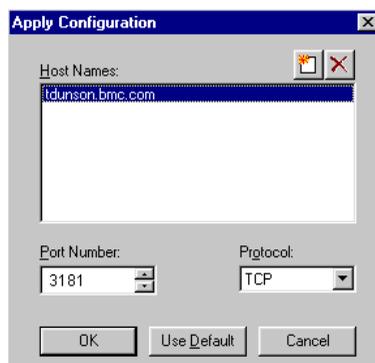
### To Apply a Remote Agent’s Configuration

**Step 1** Perform one of the following actions:

- Choose **Tools => Apply Configuration**.
- Click the Apply Configuration toolbar button.

The Apply Configuration dialog box is displayed. The dialog box is initialized using the values from the last agent communication.

**Figure 10-11 The Apply Configuration Dialog Box**



**Step 2** Specify the host name of the remote agent as well as the port and protocol to use when establishing the connection.

**Step 3** Click **OK**.

The configuration for the PATROL Agent is updated.



---

# PATROL Agent Logs

This chapter describes the many log files to which the PATROL Agent writes various information. For each type of log file, it discusses file location, contents, format, management, and aging. This chapter contains the following sections:

|  |       |
|--|-------|
| PATROL Event Log .....                                 | 11-3  |
| Events Stored .....                                    | 11-3  |
| Contents .....   | 11-3  |
| Location .....   | 11-4  |
| Setting Up the Event Log File and Size .....           | 11-4  |
| Naming the Event Log File .....                        | 11-5  |
| Setting the Event Log File Size .....                  | 11-5  |
| Managing the PATROL Event Cache .....                  | 11-6  |
| Setting the Cache Size .....                           | 11-6  |
| Setting the Number of Events Stored in the Cache ..... | 11-7  |
| Extracting Event Data from the PEM Log .....           | 11-8  |
| Before You Begin .....                                 | 11-11 |
| Syntax .....   | 11-11 |
| Options .....  | 11-12 |
| Examples .....   | 11-14 |
| Troubleshooting .....                                  | 11-14 |
| PATROL Event Archive .....                             | 11-15 |
| Contents .....   | 11-15 |
| Location .....   | 11-15 |
| Operation .....  | 11-15 |
| Select Events .....                                    | 11-16 |

|  |       |
|--|-------|
| PATROL Agent Error Log .....                               | 11-16 |
| Contents .....   | 11-16 |
| Location .....   | 11-17 |
| Sample .....   | 11-17 |
| Limiting Size By Restricting the Number of Messages.....   | 11-18 |
| Log File Aging .....                                       | 11-18 |
| Log Files for PATROL Single Sign-On for Windows 2000 ..... | 11-20 |
| Agent Audit Log .....                                      | 11-21 |
| Contents .....   | 11-22 |
| Location .....   | 11-22 |
| Setting Up Audit Logging .....                             | 11-23 |
| Keys and Values for the Audit Log Variable .....           | 11-24 |
| Audit Log File Format .....                                | 11-25 |
| Sample Audit Log File .....                                | 11-27 |

# PATROL Event Log

The PATROL Agent records events in a proprietary format. The latest events are stored in memory for quick access. This memory space is an array of structures that is called the PEM cache. As new events are created, older events are written to a log file called the PEM log. PATROL provides a utility that enables you to extract information about events from the log file.

## Events Stored

In the Standard Event Catalog, which is accessible through the console, you can determine whether an event is stored in the PEM log by setting the event's Life Expectancy. Storage options include delete if closed, delete if information, do not store, and store.

## Contents

The PATROL Agent Event log records the following information for each event:

**Table 11-1 Information Stored in Event Log**

| Information Type  | Description   |
|-------------------|---|
| event ID          | the order in which the event occurred   |
| status            | the event's current status  |
| type              | the type of event   |
| node              | the host from which the event originated  |
| origin            | the monitored object that triggered the event   |
| time and date     | the time and date that the event occurred   |
| event description | a brief description of the nature of the event  |
| event diary       | comments about what triggered this event or what type of actions were taken in response to this event<br><br>Users add these comments and the event manager time-stamps each entry. |

## Location

Table 11-2 indicates the filename and path of the event log files.

**Table 11-2 Location of Event Log Files**

| File Description | File and Path for Unix                           |
|------------------|--|
|                  | File Name and Path for Windows                   |
| event log file   | <code>\$PATROL_HOME/log/PEM_host_port.log</code> |
|                  | <code>%PATROL_HOME%\log\PEM_host_port.log</code> |

## Setting Up the Event Log File and Size

The event log file is a repository that is stored locally on computers monitored by PATROL Agents. It is not the same as the event cache, which is in memory.

When the cache becomes full, events are removed from it and placed in the repository file, which uses the following naming convention:

`PEM_hostname_port.log`

## Naming the Event Log File

The `/AgentSetup/pemLogName` configuration variable determines the naming format of the event log. The log file is stored in `PATROL_HOME\log`.

|                                |   |
|--------------------------------|---|
| <b>Format and Type of Data</b> | text string (no spaces), not applicable   |
| <b>Default Value</b>           | <code>PEM_hostname-portnum.log</code>   |
| <b>Minimum and Maximum</b>     | not applicable  |
| <b>Dependencies</b>            | none  |
| <b>Recommendation</b>          | If you change the name, use a name that indicates the log's purpose and origin. |

## Setting the Event Log File Size

The `/AgentSetup/pemLogSize` configuration variable controls the maximum size in bytes of the event log. The PEM log file is a circular file. When the file fills up, the agent deletes older events to accommodate new ones.

|                                |  |
|--------------------------------|--|
| <b>Format and Type of Data</b> | numeric, bytes   |
| <b>Default Value</b>           | 250 KB   |
| <b>Minimum and Maximum</b>     | 10240 bytes, none  |
| <b>Dependencies</b>            | none   |
| <b>Recommendation</b>          | A larger cache size improves the results in the event history, but consumes more system resources. |

# Managing the PATROL Event Cache

PATROL allows you to manage the Event Cache by size or by number of events per object. PATROL uses this variable to determine how to manage event cache size, and if by event number, how many events it should retain per object.

## Setting the Cache Size

The `/AgentSetup/pemCacheSize` configuration variable determines the size in bytes of the cache used by the agent for event management.

|                                |   |
|--------------------------------|---|
| <b>Format and Type of Data</b> | numeric, bytes  |
| <b>Default Value</b>           | 500   |
| <b>Minimum and Maximum</b>     | 10, 20480   |
| <b>Dependencies</b>            | <code>/AgentSetup/pemEvMemRetention</code> determines whether this variable has any effect. If the value of <code>pemEvMemRetention</code> is greater than zero, the PATROL Agent ignores the value of this variable. |
| <b>Recommendation</b>          | A larger cache size improves the performance of the agent but consumes more memory.<br><br>To determine the value to use, estimate 200 bytes per event.   |

## Operation

An event is triggered. The agent gets the object of the incoming event and reads the number of events stored for that object. If the number of events is less than `pemEvMemRetention`, the agent saves the incoming event in the cache. If the number is greater than or equal to the variable, the agent transfers the oldest event to the log and saves the new event in the cache. When the cache fills up, the agent increases the cache size by half of its current size. The agent bases the cache's initial size on `pemCacheSize`.

## Setting the Number of Events Stored in the Cache

The `/AgentSetup/pemEvMemRetention` configuration variable determines the number of events the PEM engine keeps in memory for each object.

|                            |  |
|----------------------------|--|
| <b>Values</b>              | <b>0</b> — <code>pemCacheSize</code> configuration variable determines the cache size used for event management<br><b><i>n</i></b> — <code>pemEvMemRetention</code> guarantees the last <i>n</i> events are retained in memory   |
| <b>Default Value</b>       | 0  |
| <b>Minimum and Maximum</b> | 0, none  |
| <b>Dependencies</b>        | <code>/AgentSetup/pemCacheSize</code> is controlled by this variable. If the value of this variable is greater than zero, the PATROL Agent ignores the value of the <code>pemCacheSize</code> variable.  |
| <b>Recommendation</b>      | Entering a value of 2 guarantees that the last two events are in memory. PATROL stores the remaining events on the local disk.<br><br>The more events stored in memory, the better the performance of the agent, but the more memory consumed. To determine the value to use, estimate 200 bytes per event.<br><br>For example, if an agent monitors 400 objects (applications, instances, parameters), the <code>pemEvMemRetention</code> is 2, and the average event size is 200 bytes, the resulting log is approximately 160K. |

### Operation

When the cache is 100% full, the agent transfers 50% of the events to the PEM log. During the transfer of events, the agent retains in cache at least one event for every object. During proper shutdown, the agent writes all events in the cache to the log.

# Extracting Event Data from the PEM Log

Use the `dump_events` command line utility to extract events from the PEM log, convert them into ASCII format, and write them to a text file.

---

**Note**

---

The `dump_events` utility does not access the PEM cache and thus does not extract the most recent events.

---

Every dumped event displays the following information by default:

- event ID
- status
- type
- date/time
- node
- origin
- event description
- event diary

The following example is an extract of PEM.txt:

---

**Example**

---

```
15 OPEN STATE_CHANGE Wed Mar 30 09:39:41 1994
palmtree CPU.CPU State Change:OK ALARM
Current state is inherited from an alert on global
parameter 'CPUCpuUtil'.
16 OPEN STATE_CHANGE Wed Mar 30 09:39:41 1994
palmtree SYSTEM.SYSTEM State Change:OK ALARM
Current state is inherited from an alert on global
parameter 'SYSCpuUtil'.
17 OPEN STATE_CHANGE Wed Mar 30 09:39:41 1994
palmtree NETWORK.NETWORK State Change:OK warn
Current state is inherited from an alert on global
parameter 'NETSGetAttr'.
18 OPEN STATE_CHANGE Wed Mar 30 09:39:42 1994
palmtree FILESYSTEM.patrol-doubloon State Change:OK
ALARM
Current state is inherited from an alert on global
parameter 'FSCapacity'.
```

---

When you use `dump_events`, keep these limitations and guidelines in mind:

- Only one copy of `dump_events` is allowed to be run at a time on a single system. Otherwise the behavior is unpredictable.
- `dump_events` should run on the same host as the PATROL Agent because both use the same lock file (`circular-file-name.lock`) in the `/$PATROL_HOME/log` directory for Unix and `PEM_hostname_port.log %PATROL_HOME%\log` directory for Windows.
- `dump_events` is installed with the agent; the PATROL environment must be set up prior to executing it.

The following directories are used by the `dump_events` utility:

**Table 11-3 dump\_events Utility Directory Structure**

| <b>Unix</b>                           | <b>Description</b>  |
|---------------------------------------|---|
| <b>Windows</b>                        |   |
| \$PATROL_HOME/log                     | The default location of the lock file, which should be non-NFS mounted. |
| %PATROL_HOME%\log                     |   |
| \$HOME/patrol                         | The default location of the ASCII file PEM.txt.                         |
| %HOMEDRIVE%\<br>%HOMEPATH%\<br>patrol |   |

## Before You Begin

The PATROL environment must be set up prior to running `dump_events`. You must set the `PATROL_HOME` environment variable. Table 11-4 describes how to define this variable for Unix and Windows.

**Table 11-4 PATROL\_HOME Environment Variable**

| Environment Variable | For Unix, run...   |
|----------------------|--|
|                      | For Windows NT, select...  |
|                      | For Windows 2000, select...  |
| PATROL_HOME          | <code>./patrolrc.sh</code> (for Korn and Bourne shell)<br><code>source .patrolrc</code> (for C shell)  |
|                      | Start => Settings => Control Panel => System => Environment; type the variable name ( <code>PATROL_HOME</code> ), the value (the path), then click Set.  |
|                      | Start => Settings => Control Panel => System => Advance, and click Environment Variables. Click New and type the variable name ( <code>PATROL_HOME</code> ), the value (the path), then, click OK. |

## Syntax

The `dump_events` utility has the following format:

```
dump_events    [-s circular-file-full-name]  
               [-d ASCII-file-full-name]  
               [-t lock-file-dir-name]  
               [-m] printf-fmt-string  
               [-f]  
               [-v]
```

# Options

The following table lists and defines the options for the `dump_events` command line utility.

**Table 11-5 Options for the `dump_events` Command Line Utility (Part 1 of 2)**

| Option | Definition  |
|--------|---|
| -s     | This option provides the full name of the event log file (circular file) if different than the default.<br><br>The default is<br>\$PATROL_HOME/log/PEM_host-3181.log. for Unix and<br>%PATROL_HOME%\log\PEM_host-3181.log. for<br>Windows. If you do not provide a directory, the default<br>directory is used. |
| -d     | This option provides the full name of the ASCII dump file if different than the default. The default is<br>\$HOME/patrol/PEM_host-3181.txt. If you do not<br>provide a directory, the default directory is used.  |
| -t     | This option provides the directory name of the circular file lock. The default is \$PATROL_HOME/log for Unix and<br>%PATROL_HOME%\log for Windows. If you do not provide<br>a directory, the default directory is used.   |
| -f     | This option enables the event-dumping and can be used if a<br>previous <code>dump_events</code> program was stopped.  |

**Table 11-5 Options for the dump\_events Command Line Utility  
(Part 2 of 2)**

| Option | Definition   |
|--------|--|
| -m     | <p>This option provides a user-defined format for event dump. By default, the event manager uses the print format string equivalent to</p> <pre data-bbox="588 361 1216 387">\ %1\$s %2\$s %3\$s %4\$s %5\$s %6\$s %7\$s %8\$s\n'.</pre> <p>The arguments have the following order:</p> <ol data-bbox="588 447 790 682" style="list-style-type: none"> <li>1. Event ID (%1\$s)</li> <li>2. Status (%2\$s)</li> <li>3. Type (%3\$s)</li> <li>4. Time (%4\$s)</li> <li>5. Node (%5\$s)</li> <li>6. Origin (%6\$s)</li> <li>7. Desc (%7\$s)</li> <li>8. Diary (%8\$s)</li> </ol> <p>The following escape sequence is recognized in the <i>printf-fmt-string</i>:</p> <ul data-bbox="588 777 768 890" style="list-style-type: none"> <li>\n new line</li> <li>\v vertical tab</li> <li>\t tab</li> <li>\r carriage return</li> </ul> |
| -v     | Displays version information.  |

## Examples

```
dump_events -m  
'%8$s %7$s %6$s %5$s %4$s %4$s %3$s %2$s %1$s\n'
```

Note the single quotes around the string format. This command dumps event fields in reverse order to the default, as follows:

1. Diary
2. Description
3. Origin
4. Node
5. Time
6. Type
7. Status
8. Event ID

```
dump_events -m '\t%s \t%s \t%s \t%s \t%s \t%s  
\t%s \t%s\n\n'
```

This command dumps fields in the default order, separated by tabs. Three new lines separate events.

## Troubleshooting

If the `dump_events` utility fails, check the following items:

- The files `PEM_host-3181.log` and `PEM_host-3181.log.lock` on Unix and `PEM_host-3181.log-lock` on Windows have the correct file privileges.
- Your environment variables `$HOME`, `$PATROL_HOME`, and `$PATROL_ADMIN` are correct.
- Your lock file directory is not NFS mounted.

# PATROL Event Archive

PATROL provides a separate facility, the PEM Archive file, to record and store events based upon a user-defined filter. The archive depends upon the PEM cache. The agent uses the cache size to determine when to write to the archive. However, the PEM archive files and PEM log file are mutually exclusive.

## Contents

The PATROL Agent Event archive records the same information as the PEM log. For a description of the logs contents, see “Contents” on page 11-3.

## Location

Table 11-6 indicates the filename and path of the event log files.

**Table 11-6 Location of Event Archive Files**

| File Description   | File and Path for Unix                               |
|--------------------|--|
|                    | File Name and Path for Windows                       |
| event archive file | <code>\$PATROL_HOME/log/PEM_host_port.archive</code> |
|                    | <code>%PATROL_HOME%\log\PEM_host_port.archive</code> |

## Operation

Every time the cache gets half full (`pemCacheSize/2`), the PATROL Agent generates an `EventArchive` event. This event class’s notification command scans the cache and writes certain events (based on a user-defined filter) to a PEM archive file.

## Select Events

You can specify which events are written to the archive file in the EventArchive class of the Standard Event Catalog. For information on how to use the event\_archive() function and how to build a filter for it, see PATROL Script Language *Reference Manual*.

## PATROL Agent Error Log

As part of the PATROL Agent's many tasks, it writes messages to the PATROL Agent error log file. Messages sent to the error log include both error messages that result from a failed action and informational messages that result from successful action. The error log is not exhaustive and does not record the success or failure of every action.

## Contents

The PATROL Agent Error log records the following information for each error message.

**Table 11-7 Information Stored in PATROL Error Log**

| Information Type | Description  |
|------------------|--|
| error message ID | the catalog message ID issued by the agent<br><br>The catalogs are stored in <code>\$PATROL_HOME/lib/nls/C/##.cat</code> files. If an error message is truncated due to a crash, you can use the ID to identify the last message written to the error log. |
| time and date    | the time and date that the event occurred  |
| error message    | the message written to the error log<br><br>Messages written to the error log are not necessarily the result of errors or failures. Some PATROL components send informational messages to the error log.   |

## Location

Table 11-8 indicates the filename and path for error log files.

**Table 11-8 Location of PATROL Agent Error Log File**

| File Description            | File and Path for Unix                         |
|-----------------------------|--|
|                             | File Name and Path for Windows                 |
| PATROL Agent error log file | <code>\$PATROL_HOME/log/PatrolAgent.err</code> |
|                             | <code>%PATROL_HOME%/log/PatrolAgent.err</code> |

## Sample

The following sample has been extracted from the beginning of a PATROL Agent Error Log file.

```
PatrolAgent (V3.4.00.0, Windows NT 3.51 x86, GA_3_3_0-23-19990407 Apr
8 1999)
ID 142 started at Sat Apr 01 12:28:59 CCYY
Host dliscum__nt WindowsNT dliscum__nt 4.0 1381-5 Intel/686
ID 102043: Sat Apr 01 12:29:00 CCYY: Found license for 10 agents
ID 102071: Sat Apr 01 12:29:05 CCYY: Patrol SNMP Sub-Agent connection
failed - Make sure SNMP Master Agent is running.
ID 102012: Sat Apr 01 12:29:13 CCYY: Binding PatrolAgent to UDP port
3939
ID 102010: Sat Apr 01 12:29:13 CCYY: Binding PatrolAgent to TCP port
3939
ID 102097: Sat Apr 01 12:29:17 CCYY: COM: com library initialize OK,
COM security setting is N.
ID 10205a: Sat Apr 01 12:29:17 CCYY: Binding PatrolAgent to port 3939
...
ID 10209d: Sat Apr 01 12:31:10 CCYY: SNMP Sub-Agent is now connected.
ID 10200a: Sat Apr 01 17:25:23 CCYY: Console U:8562.1040@127.0.0.1
logged in as user PATROL_AD2 (type Developer).
ID 10209e: Sat Apr 01 17:25:40 CCYY: SNMP Sub-Agent is now
disconnected.
ID 102002: Sat Apr 01 17:25:40 CCYY: Unknown appl 'PATROL_NT' -
nothing will be sent to the Console
```

## Limiting Size By Restricting the Number of Messages

The `/AgentSetup/maxAgentMessageLimit` configuration variable determines the number of messages written to the PATROL Agent error log.

|                                |  |
|--------------------------------|--|
| <b>Format and Type of Data</b> | numeric, messages  |
| <b>Default Value</b>           | 100000   |
| <b>Minimum and Maximum</b>     | 0, none  |
| <b>Dependencies</b>            | none   |
| <b>Recommendation</b>          | <p>Under normal circumstances, the default value for the number of messages should not be reached. Most of the PATROL Agent messages are a single line long. Only a few messages exceed this size.</p> <p>The global symbol <code>numAgentMsgs</code> tracks how many messages have been written to the log.</p> |

You can limit the number of messages that are written to the agent error log using the agent configuration variable, `maxAgentMessageLimit`. When the limit is reached, the following block is written to the agent error log:

```
Mon Aug 14 11:45:22 2000 Maximum number of messages  
(12) logged. >>>NO MORE MESSAGES WILL BE LOGGED<<<
```

## Log File Aging

The PATROL Agent retains the current error log and the five most recent ones. To create an error log archive, the agent appends the extension `~#~` to the error log, where `#` indicates the log's relative age. One (1) represents the newest archived log and five (5) represents the oldest.

## Example

These files illustrate the PATROL Agent Error Log file aging technique.

```
PatrolAgent-hostname-port.errs  
PatrolAgent-hostname-port.errs.~1~  
PatrolAgent-hostname-port.errs.~2~  
PatrolAgent-hostname-port.errs.~3~  
PatrolAgent-hostname-port.errs.~4~  
PatrolAgent-hostname-port.errs.~5~
```

## Operation

During startup, the PATROL Agent checks to see if a log file or any archived log files exist.

1. If a fifth archive log exists, the agent deletes that log.
2. If archives one through four exist, the agent increments the extension of each by one.
3. If an error log file exists, the agent archives it by adding the extension ~1~.
4. The agent opens a new error log file.

# Log Files for PATROL Single Sign-On for Windows 2000

Single Sign-On does not have a separate log facility. When the PATROL Agent is in debug mode, the agent writes user log-on validation and rejection messages to the agent log file.

# Agent Audit Log

The Audit Log feature records various security-related aspects of PATROL. The Log records information such as:

- commands that are executed as a result of Infobox or Menu commands
- which console-connection runs commands (listed by console ID)
- connect/disconnect
- commit operations
- configuration operations
- most spawned commands

---

**Tip**

---

This information is critical for locating the source of PAWorkRateExecsMin alarms.

---

# Contents

Table 11-9 lists the type of information recorded in the audit log file.

**Table 11-9 Type of Information Recorded in the Audit Log**

| <b>Type of Information/Event</b> | <b>The Audit Log Records...</b>   |
|----------------------------------|---|
| Spawned Commands                 | <p>explicitly created external processes.</p> <p><b>Note:</b> The agent does not create a log entry for implicitly created commands. This means that the PATROL agent will not log the commands that are created by a process that it creates.</p> <p><b>Example:</b> Using PSL popen() to create a process, and then sending a command down the channel for this process to execute. The agent logs the creation of the popen() process.</p> |
| Commands Executed                | <p>each command (i.e. script) that is executed as a result of a Menu Command or an InfoBox Command.</p> <p>The entry in the log file records the console ID of the peer and the local account name used for the connection.</p>   |
| Connect/Disconnect Details       | <p>each connection/disconnection.</p> <p>The entry in the log file records the console ID of the peer, the console type, and the local account name used for this connection.</p>   |
| Commit Actions                   | <p>each file that is transferred during a commit.</p> <p>The entry in the log file records the name of the file, the console ID of the connection performing the commit, and the local account that is used for the connection.</p>   |
| Configuration Actions            | <p>each explicit pconfig, wpconfig, or xpconfig action that affects the state of the PATROL Agent.</p> <p>The entry in the log file records the events that change variables, kill the agent, and send a license file and PSL pconfig() operations.</p>   |

## Location

The file path and name are user-defined. You determine the file location during audit logging setup.

# Setting Up Audit Logging

The auditing feature is controlled by the configuration variable `/AgentSetup/auditLog`. The standard PATROL installation process does not create this variable. You must create and set this variable to enable audit logging.

The standard PATROL installation process does not create this variable. You must create and set this variable to restrict the number of messages.

|                                |  |
|--------------------------------|--|
| <b>Format and Type of Data</b> | see “Keys and Values for the Audit Log Variable” on page 11-24 |
| <b>Default Value</b>           | none   |
| <b>Minimum and Maximum</b>     | not applicable   |
| <b>Dependencies</b>            | none   |
| <b>Recommendation</b>          | none   |

Figure 11-1 demonstrates what the variable would look like when added through `wpconfig` or `xpconfig`.

**Figure 11-1 Adding `/AgentSetup/auditLog` in `wpconfig`/`xpconfig`**



## Keys and Values for the Audit Log Variable

The Audit Log configuration variable, `/AgentSetup/auditLog`, consists of a new line separated list of `KEY=VALUE` pairs.

**Table 11-10 AgentSetup/auditLog Keys and Values**

| Key       | Description   |
|-----------|---|
| Active    | <p>determines whether the audit logging feature is turned on or not. The recognized values include:</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> - turns on audit logging; other valid values that can be used are <b>On</b>, <b>True</b>, and <b>1</b></li> <li>• <b>No</b> - turns off audit logging; default setting; other valid values that can be used are <b>No</b>, <b>False</b>, and <b>0</b></li> </ul>  |
| Delimiter | <p>determines the delimiter character that separates the fields in the log file. The default character is the pipe-symbol ' '.</p>  |
| FileAging | <p>determines the interval at which a new log file is created.</p> <ul style="list-style-type: none"> <li>• <b>Daily N</b> - create a new log file every day at approximately the hour <i>N</i>, where <i>N</i> ranges from midnight 12 a.m. represented as 0 to 11 p.m. represented as 23; the default is Daily 0</li> <li>• <b>Entries N</b> - create a new log file after logging <i>N</i> entries, where <i>N</i> is the number of entries; for example, <i>N</i> &gt;= 100</li> <li>• <b>Size N</b> - create a new log file when the file reaches a designated size, where <i>N</i> is the file size in KB; for example, <i>N</i> &gt;= 32</li> </ul>  |
| FileCount | <p>determines how many old log files are retained. The default value is 5</p> <p>Each time a new log-file is created, the previous files are renamed in the same manner as done with the agent regular log-file.</p>  |
| FileName  | <p>determines the pathname and filenaming convention for the audit log file. The name can contain the following macros</p> <ul style="list-style-type: none"> <li>• <b>%H</b> - refers to the current agent-host</li> <li>• <b>%P</b> - refer to the port-number being used</li> </ul> <p>If path is not a fully qualified pathname, the PATROL Agent treats it as being relative to the <code>PATROL_HOME/log</code> directory. All subdirectories in the pathname must already exist. PATROL Agent creates the log file but not the directories leading up to the file. If the file cannot be opened, the agent writes an error message to the agent's log file.</p> <p>The default path and file name is</p> <p><b>NT</b>—%PATROL_HOME%\log\PatrolAgent-%H-%P.audit<br/> <b>Unix</b>—\$PATROL_HOME/log/PatrolAgent-%H-%P.audit</p> |

# Audit Log File Format

The log file stores data in the following format:

```
Time | Host | EntryType | User | Entry-specific-data
```

Each field is separated by the delimiter character (the default is a pipe, |) specified in `/AgentSetup/auditLog` configuration variable.

**Table 11-11 Audit Log File Format**

| <b>Field</b> | <b>Description</b>   |
|--------------|--|
| Time         | the date and local time in <b>yyymmdd:hh:mm:ss</b> format  |
| Host         | the name of the machine on which the agent is running  |
| EntryType    | <p>the type of action being recorded</p> <ul style="list-style-type: none"><li>• audit</li><li>• execute</li><li>• connect</li><li>• disconnect</li><li>• commit</li><li>• config</li><li>• command</li></ul> <p>Entry- specific data field description provides details on what type of information each entry type provides.</p> |
| User         | the name of the local account used to perform the action   |

**Table 11-11 Audit Log File Format**

| <b>Field</b>         | <b>Description</b>  |  |
|----------------------|---|--|
| Entry- specific data | The Entry Type is determined by the type of action being recorded. The left column lists the action; the right describes the entry. |  |
|                      | audit   | Indicates file opened/closed   |
|                      | command   | the console ID running the command; if the command originates from the system-output window, it displays the actual command  |
|                      | commit  | the console ID and the name of the file being transferred  |
|                      | config  | two types of entries <ul style="list-style-type: none"><li>• The first indicates where the connection originated. It contains the console ID and the high-level action taking place such as reboot agent.</li><li>• The second gives a specific action such as store or delete, and lists the variable affected.</li></ul> |
|                      | connect   | the console ID and the connection type   |
|                      | disconnect  | the console ID of the connection   |
|                      | execute   | the command name and its arguments   |

# Sample Audit Log File

```
CCYY0528:15:18:45|PAYROLL_NT4|audit|PatrolAgent|File opened
CCYY0528:15:18:47|PAYROLL_NT4|execute|PAYROLL_ADM2| /bin/ps -elf
CCYY0528:15:18:54|PAYROLL_NT4|execute|PAYROLL_ADM2| /bin/ksh -c uname
-a
CCYY0528:15:18:56|PAYROLL_NT4|connect|PAYROLL_ADM2|U:7412.51622@172.1
9.205.24 Developer
CCYY0528:15:18:56|PAYROLL_NT4|execute|PAYROLL_ADM2| /bin/ksh -c
/bin/ksh
CCYY0528:15:18:56|PAYROLL_NT4|execute|PAYROLL_ADM2| /bin/ksh -c
/bin/ksh
CCYY0528:15:19:04|PAYROLL_NT4|execute|PAYROLL_ADM2| /bin/ksh -c mount
-v
CCYY0528:15:19:04|PAYROLL_NT4|command|PAYROLL_ADM2|U:7412.51622@172.1
9.205.24
CCYY0528:15:19:04|PAYROLL_NT4|command|PAYROLL_ADM2|U:7412.51622@172.1
9.205.24
CCYY0528:15:19:04|PAYROLL_NT4|execute|PAYROLL_ADM2| /bin/ksh -c
nfsstat
CCYY0528:15:19:04|PAYROLL_NT4|command|PAYROLL_ADM2|U:7412.51622@172.1
9.205.24
CCYY0528:15:19:04|PAYROLL_NT4|execute|PAYROLL_ADM2| /bin/ksh -c file
/etc/utmp
CCYY0528:15:19:04|PAYROLL_NT4|execute|PAYROLL_ADM2| /bin/ksh -c sh -c
'test -r /etc/utmp ;echo $?'
CCYY0528:15:19:04|PAYROLL_NT4|execute|PAYROLL_ADM2| /bin/ksh -c sh -c
'test -f /etc/utmp ;echo $?'
CCYY0528:15:19:04|PAYROLL_NT4|pconfig|<>|Store
/LOG/files/etc-utmp/filter
CCYY0528:15:19:04|PAYROLL_NT4|pconfig|<>|Store
/LOG/files/etc-utmp/path
CCYY0528:15:19:04|PAYROLL_NT4|pconfig|<>|Store
/LOG/files/etc-utmp/dump
CCYY0528:15:19:04|PAYROLL_NT4|pconfig|<>|Store
/LOG/files/etc-utmp/fpos
...
CCYY0528:15:20:15|PAYROLL_NT4|command|PAYROLL_ADM2|U:7412.51622@172.1
9.205.24 Agent KILL command
CCYY0528:15:20:15|PAYROLL_NT4|disconnect|PAYROLL_ADM2|U:7412.51622@17
2.19.205.24
CCYY0528:15:20:20|PAYROLL_NT4|audit|PatrolAgent|File closed
```



---

# PATROL Agent Parameter History

This chapter describes the PATROL Agent Parameter History Databases. It discusses the types of information recorded in the history databases, the database structure, and the location of the databases in the PATROL directory structure. It also provides information on how to extract history information from the database into ASCII text format and repair corrupted history databases. This chapter contains the following sections:

|   |       |
|---|-------|
| Overview of PATROL History . . . . .  | 12-3  |
| Contents . . . . .  | 12-3  |
| Structure . . . . .   | 12-4  |
| Location . . . . .  | 12-4  |
| Setting Up the History Cache and Database. . . . .                                      | 12-6  |
| Flushing the Agent Cache Based on a Time interval. . . . .                              | 12-7  |
| Flushing the Agent Cache Based on the Number of<br>Data Points . . . . .                | 12-7  |
| Defining How Much Time the Agent Spends Writing Cache to a<br>History File . . . . .    | 12-8  |
| Setting the Number of Days that History is Retained in the<br>History Database. . . . . | 12-8  |
| Extracting History from the Database . . . . .  | 12-10 |
| Output . . . . .  | 12-10 |
| File and Directory Structure Used by dump_hist . . . . .                                | 12-11 |
| Before Running the dump_hist Utility. . . . .   | 12-13 |
| Running dump_hist . . . . .   | 12-13 |
| Examples . . . . .  | 12-18 |

|   |       |
|---|-------|
| Fixing a Corrupted History Database .....           | 12-20 |
| Output .....  | 12-21 |
| File and Directory Structure Used by fix_hist ..... | 12-22 |
| Before Running the fix_hist Utility .....           | 12-24 |
| Running fix_hist .....                              | 12-24 |
| Examples .....                                      | 12-27 |

# Overview of PATROL History

The PATROL Agent records the values of each parameter and stores them in a proprietary parameter history database. This database remains open while the agent is running. If the agent does not shut down properly, the database may be corrupted. PATROL provides a utility that enables you to extract information from the database about a parameter based on its class, instance, and time period. It also provides a utility for repairing corrupted history databases.

## Contents

The PATROL Agent History database records for each data point the following information:

**Table 12-1 Information Stored in History Database**

| <b>Information Type</b>      | <b>Description</b>  |
|------------------------------|---|
| Port number                  | port number on which the agent listens                    |
| Host name                    | name of the host on which the agent is running            |
| Class                        | application class to which the data point belongs         |
| Instance                     | application instance to which the data point belongs      |
| Parameter                    | parameter to which the data point belongs                 |
| Time and date                | day, time, and date on which the data point was collected |
| Annotated text (if provided) | descriptive text associated with the data point           |

## Structure

PATROL stores history information in a history database that consists of three separate files:

- index file—points to the location of the data in the database
- history text file—contains annotations that a user has added to a data point
- history data file—contains data points

The history database stores information in a binary format. To view the information, you must use the PATROL Console or run the `dump_hist` utility.

## Location

The PATROL Agent provides environment variables that enable you to determine where the agent creates the history database. You can designate a location in or out of the PATROL directory structure or on another host within the network. In addition, you can establish a location by port number. This feature allows you to accommodate a host with multiple agents by directing each agent running on a unique port number to write to a different history database.

### Custom Location

The environment variables that allow you to specify the location of an agent's history database are

- `PATROL_HISTORY`
- `PATROL_HISTORY_PORT`
- `PATROL_VIRTUALNAME`

For information on how to use these environment variables, see “Variables” on page 6-28.

## Default Location

Table 12-2 indicates the filename and path of files that compose the history database.

**Table 12-2 Location of History Files**

| File Description  | Filename and Path for Unix                                      |
|-------------------|---|
|                   | Filename and Path for Windows                                   |
| Index file        | <i>\$PATROL_HOME/log/history/hostname/port_num/dir</i>          |
|                   | <i>%PATROL_HOME%\log\history\hostname\port_num\dir</i>          |
| History text file | <i>\$PATROL_HOME/log/history/hostname/port_num/annotate.dat</i> |
|                   | <i>%PATROL_HOME%\log\history\hostname\port_num\annotate.dat</i> |
| History data file | <i>\$PATROL_HOME/log/history/hostname/port_num/param.hist</i>   |
|                   | <i>%PATROL_HOME%\log\history\hostname\port_num\param.hist</i>   |

:

# Setting Up the History Cache and Database

PATROL provides two different ways of controlling how much parameter information, in the form of data points, that it stores in memory. The memory storage is referred to as the history cache. You can retain information in the history cache based on either how much information you want to store (number of data points), or how long you want to store information in memory.

These two storage methods are not mutually exclusive.

- If you choose to store history based on the number of data points, the agent writes all the data points for a parameter to the history database when that parameter collects the specified number of data points.
- If you choose to store history based on a time interval, when the interval expires, the agent writes all data points for all parameters to the history database.
- If you activate both methods of storage, the agent writes information to the history database more frequently than if only one method is activated. The agent writes once when the time interval expires and once when a parameter collects the specified number of data points.

---

**Note**

---

This history is not the same as events, which are kept in the event log repository file.

---

## Flushing the Agent Cache Based on a Time interval

The `/AgentSetup/prmHistCacheFlushTimer` configuration variable determines the period of time that the cache is flushed to the history database. History information that is stored in the cache can be accessed through the PATROL Console. Once the history information is written to the history database, it can still be accessed through the console.

|                                |                   |
|--------------------------------|-------------------|
| <b>Format and Type of Data</b> | numeric, seconds  |
| <b>Default Value</b>           | 1200 (20 minutes) |
| <b>Minimum and Maximum</b>     | 5, none           |
| <b>Dependencies</b>            | none              |
| <b>Recommendation</b>          | none              |

## Flushing the Agent Cache Based on the Number of Data Points

The `/AgentSetup/prmHistCacheSize` configuration variable sets the number of data points allowed in the cache, for a specific parameter, before writing the cache to the history file. For example, if you set the value of `prmHistCacheSize` to 10, when the agent has collected 10 data points for a parameter it writes the data to the history file.

To turn off history cache and thus turn off recording history, set this variable equal to or less than 1.

|                                |                                    |
|--------------------------------|------------------------------------|
| <b>Format and Type of Data</b> | numeric, data points per parameter |
| <b>Default Value</b>           | 16                                 |
| <b>Minimum and Maximum</b>     | 0, none                            |

|                       |      |
|-----------------------|------|
| <b>Dependencies</b>   | none |
| <b>Recommendation</b> | none |

## Defining How Much Time the Agent Spends Writing Cache to a History File

You can use the `/AgentSetup/prmHistCacheMaxFlushTime` configuration variable to define how much time the agent spends writing cache data to a history file. For example, if `prmHistCacheMaxFlushTime` is set to 600 seconds, the agent stops writing cache data to the history file after 600, seconds even if it has not written all the cache data to the history file.

|                                |   |
|--------------------------------|---|
| <b>Format and Type of Data</b> | numeric, seconds  |
| <b>Default Value</b>           | 600   |
| <b>Minimum and Maximum</b>     | 1 is minimum<br>0 flushes all data without a time limit |
| <b>Dependencies</b>            | none  |
| <b>Recommendation</b>          | none  |

## Setting the Number of Days that History is Retained in the History Database

The `/AgentSetup/historyRetentionPeriod` configuration variable determines the number of days that collected parameter values are retained in the history database. After this retention period has expired, PATROL deletes the retained information.

|                                |               |
|--------------------------------|---------------|
| <b>Format and Type of Data</b> | numeric, days |
| <b>Default Value</b>           | 1             |

|                            |   |
|----------------------------|---|
| <b>Minimum and Maximum</b> | 0, none   |
| <b>Override</b>            | history variable at the application or parameter level  |
| <b>Dependencies</b>        | none  |
| <b>Recommendation</b>      | To retain historical data, set this value to coincide with weekly (7) or monthly (30) backups. At the end of each period, shut down the agent, save the history database files to a location outside of the PATROL directory structure, and then restart the agent. |

## From Console

For information on how to set the history retention period at the computer instance, application class, application instance, or parameter level, see *PATROL Console for Microsoft Windows 2000 User Guide, Volume II—Monitoring and Managing with PATROL* or *PATROL Console for Unix User Guide*.

# Extracting History from the Database

Use the `dump_hist` command line utility to extract parameter history data from databases maintained by PATROL Agents and save the data to ASCII text files. History data is printed to standard output in text form.

## Output

The `dump_hist` utility extracts the following information by default:

- hostname
- application class
- instance
- parameter
- time and date
- value

The following example is a portion of a history dump output.

---

**Example**

---

```
HOST_12/NT_CACHE.NT_CACHE/CACcachCopyReadHitsPercent
  Fri May 05 09:22:56 CCYY 98.5958
  Fri May 05 09:23:56 CCYY 98.324
  Fri May 05 09:24:56 CCYY 100
  Fri May 05 09:25:57 CCYY 99.4318
  Fri May 05 09:26:57 CCYY 100
  Fri May 05 09:27:57 CCYY 100
  Fri May 05 09:28:57 CCYY 90.1566
  Fri May 05 09:29:57 CCYY 96.1365
  Fri May 05 09:30:57 CCYY 92.2062
```

---

# File and Directory Structure Used by dump\_hist

PATROL creates and maintains a standard file and directory structure in which it stores the history database. The dump\_hist utility will not run if the proper file and directory structure do not exist.

## History Database Directory Structure

Table 12-3 lists the directories for dump\_hist.

**Table 12-3 dump\_hist Directories**

| <b>Unix Directory</b>                              | <b>Definition</b>   |
|--|---|
| <b>Windows Directory</b>                           |   |
| <i>\$PATROL_HOME</i>                               | directory specified when the PATROL Agent is started  |
| <i>%PATROL_HOME%</i>                               |   |
| <i>\$PATROL_HOME/log</i>                           | log directory for the PATROL Agent  |
| <i>%PATROL_HOME%\log</i>                           |   |
| <i>\$PATROL_HOME/log/history</i>                   | top directory of parameter history databases contains top directories of each PATROL Agent  |
| <i>%PATROL_HOME%\log\history</i>                   |   |
| <i>\$PATROL_HOME/log/history/hostname</i>          | a PATROL Agent's top directory of parameter history created by the PATROL Agent using the official host name on which it is running                               |
| <i>%PATROL_HOME%\log\history\hostname</i>          |   |
| <i>\$PATROL_HOME/log/history/hostname/port_num</i> | the directory where a single PATROL Agent stores files when multiple PATROL Agents are running on the same host created by the PATROL Agent using its port number |
| <i>%PATROL_HOME%\log\history\hostname\port_num</i> |   |

## History Database File Composition

Table 12-4 shows the files read by dump\_hist.

**Table 12-4 Files Read by dump\_hist**

| File Description  | Filename and Path for Unix   |
|-------------------|--|
|                   | Filename and Path for Windows  |
| Index file        | <i>\$PATROL_HOME</i> /log/history/ <i>hostname</i> / <i>port_num</i> /dir            |
|                   | % <i>PATROL_HOME</i> %\log\history\ <i>hostname</i> \ <i>port_num</i> \dir           |
| History text file | <i>\$PATROL_HOME</i> /log/history/ <i>hostname</i> / <i>port_num</i> /annotate.dat   |
|                   | % <i>PATROL_HOME</i> %\log\history\ <i>hostname</i> \ <i>port_num</i> \annotate.dat  |
| History data file | <i>\$PATROL_HOME</i> /log/history/{ <i>hostname</i> }/{ <i>port_num</i> }/param.hist |
|                   | % <i>PATROL_HOME</i> %\log\history\ <i>hostname</i> \ <i>port_num</i> \param.hist    |

## Before Running the dump\_hist Utility

The PATROL environment must be set up prior to running dump\_hist. You must set the PATROL\_HOME environment variable.

Table 12-5 describes how to define this variable for Unix and Windows.

**Table 12-5 PATROL\_HOME Environment Variable**

| Environment Variable | For Unix, run...  |
|----------------------|---|
|                      | For Windows NT, select...   |
|                      | For Windows 2000, select...   |
| PATROL_HOME          | <p>./patrolrc.sh (for Korn and Bourne shell)<br/>source .patrolrc (for C shell)</p> <p><b>Start =&gt; Settings =&gt; Control Panel =&gt; System =&gt; Environment</b>; type in the variable name (PATROL_HOME) and value (the path), and click Set.</p> <p><b>Start =&gt; Settings =&gt; Control Panel =&gt; System =&gt; Advance</b>, and click Environment Variables. Click New and type in the variable name (PATROL_HOME) and value (the path). Then, click OK.</p> |

## Running dump\_hist

---

### Note

---

Running dump\_hist while the PATROL Agent is also running can cause unpredictable results.

---

- » Type the following command at the command line prompt and press **Enter**:

```
dump_hist -options
```

## Syntax

The `dump_hist` utility has the following formats:

```
dump_hist    [-p port]
              [-host hostname]
              [-class regexp]
              [-inst regexp]
              [-format OPTIONS format]
              [-param regexp]
              [-s mmddhhmm[yyyy]]
              [-e mmddhhmm[yyyy]]
              [-timestamp]
              [-annotate]
              [-nostat]
              [-v]
```

## Return Values

The utility exits with one of the following values:

- 0 All matching parameters were dumped successfully
- 1 `dump_hist` was aborted because of invalid options

## Options

Table 12-6 lists and defines the options recognized by the `dump_hist` utility.

**Table 12-6** `dump_hist` Utility Options (Part 1 of 2)

| Option            | Definition  |
|-------------------|---|
| -p                | This option specifies which PATROL Agent's history database is used. Combined with the <code>-host</code> option, <code>-p</code> specifies a unique PATROL Agent. The value of this option should be the port number to which the specified PATROL Agent is listening. The default value is 3181.                              |
| -host             | This option specifies which PATROL Agent's history database is used. By default, the name of the host on which <code>dump_hist</code> is started is used.   |
| -class            | This option specifies which application classes to dump. You can use <code>-class</code> to filter out parameters whose application classes do not match the given regular expression.  |
| -inst             | This option specifies which application instances to dump. You can use <code>-inst</code> to filter out parameters whose application instances do not match the given regular expression.   |
| -format "options" | See "Format Options" on page 12-16.   |
| -param            | This option specifies which parameters to dump. You can use <code>-param</code> to filter out parameters whose names do not match the given regular expression.   |
| -s<br>-e          | These options specify the time span for which you want to view history. Only parameter points that occur after the start date\time ( <code>-s mmddhhmm[yyyy]</code> ) and before the end date\time ( <code>-e mmddhhmm[yyyy]</code> ) are provided.<br><br>See the example entitled "For a Specific Time Period" on page 12-19. |

**Table 12-6 dump\_hist Utility Options (Part 2 of 2)**

| Option     | Definition   |
|------------|--|
| -timestamp | This option prints time stamps as integers instead of converting them into a local time string.<br><br>See the example entitled "To Print Time as an Integer Rather than a Local Time String" on page 12-18. |
| -annotate  | This option prints annotation text for a data point that is annotated.   |
| -nostat    | This option suppresses the display of summary statistics. Only the raw history data is printed, which simplifies parsing.  |
| -v         | This option displays version information.  |

If multiple options are specified, they are interpreted as "and" conditions. For example, if you specify both -class and -param, then for the parameter to be dumped, its application class must match the regular expression given by the -class option, and its application instance must match the regular expression given by the -inst option.

### Format Options

The -format option controls the layout of ASCII text files created by the dump\_hist command. This option can be used to decide what information is in the file. It can also make these files easier to read or import into popular desktop applications such as spreadsheet and word processing programs. Using a PATROL KM for history loading, it can facilitate uploading these files into databases such as Oracle, Sybase, and Microsoft SQL Server. (See the *PATROL History Loader Knowledge Module User Guide* for more information.)

**Table 12-7 Options for -format of dump\_hist (Part 1 of 2)**

| Option | Definition       |
|--------|------------------|
| %H     | host name        |
| %A     | application name |
| %I     | instance name    |
| %P     | parameter name   |

**Table 12-7 Options for -format of dump\_hist (Part 2 of 2)**

| <b>Option</b> | <b>Definition</b>                                |
|---------------|--|
| %y            | year (yyyy)                                      |
| %m            | month (mm)                                       |
| %d            | day (dd)   |
| %h            | hour   |
| %M            | minutes (MM)                                     |
| %s            | seconds (ss)                                     |
| %v            | parameter value stored in <b>param.hist</b> file |

The syntax of -format requires placing the options in Table 12-7 between quotes and using characters such as commas (,), pipes (|), and dashes (-) to separate these values. Consider the following example:

```
dump_hist -format ", ,%H,%A,%I,%P,%Y-%m-%d %h:%M:%s,%v"
```

This command produces a **.dat** file in the **\$PATROL\_HOME/remote** directory on Unix and **%PATROL\_HOME%\remote** directory on Windows, and each line of the file uses the same character separators.

Following is a sample line from a file generated by this example:

```
, ,smile,USERS,USERS,USRNoSession,CCYY-10-16  
20:51:53,22
```

## Examples

The following examples illustrate how to use the dump history utility to extract certain types of information.

### For Current Host

This command prints all parameters for the current host.

```
dump_hist
```

### For Application Instance

This command prints all parameters of the application instance, `db_manager`, on a host entitled `host1`.

```
dump_hist -host host1 -p 3181 -inst db_manager
```

### To Print Time as an Integer Rather than a Local Time String

This command disallows the conversion of time stamps to a local time string.

```
dump_hist -host host1 -p 3181 -inst db_manager  
-timestamp
```

Following is a sample line of output generated by this command:

```
smile|USRNoSession|22|10/16/CCYY|202153
```

### To Write to a File

This command redirects the output of `dump_hist` to a file called **dump\_hist.out**. Each line of this file uses a pipe symbol to separate the values and displays the host name, parameter name, parameter value, date, and time.

```
dump_hist -format "%H|%P|%v|%m/%d/%y|%h%M%S"  
>dump_hist.out
```

## For a Specific Time Period

This command prints history data from October 1, CCYY 8:00AM through October 1, 9:00AM of the same year.

```
dump_hist -s 10010800CCYY -e 10010900CCYY
```

---

### Note

---

CCYY represent the century and year. When using this format option, enter the actual year.

---

# Fixing a Corrupted History Database

Problems with the history database sometimes develop through undesirable incidents such as an abrupt shut down of the PATROL Agent. In these cases, one or more pointers in the **dir** file could lead to erroneous data in the other two files. Corrupted history can cause the agent to consume excessive amounts of CPU and memory.

The `fix_hist` command tries to solve these problems by removing the errant pointers and packing the related files: **dir**, **annotate.dat**, and **param.hist**. Often you will find out that your history database has these problems when you see an error message similar to the following:

```
Found inconsistency in hist param database.
```

---

## Note

---

The history index file **does not** have a file size limit.

---

Once such a problem is discovered, you may want to perform one of the following actions:

- Shut down the PATROL Agent and run the `fix_hist` utility, and remove any history you no longer need by choosing **Options => Clear History** from the parameter list screen.
- Shut down the PATROL Agent and move the **annotate.dat**, **dir**, and **param.hist** files.

Be sure to move or delete all three files. When the agent is restarted, it creates new versions of these files.

## Output

While reading the annotation database, the `fix_hist` utility creates a file. When the utility encounters an error that it cannot repair, it saves the data that it has already read and discards the data that is unreachable due to a corrupted pointer or other error.

The fix history utility backs up the history database files that it attempts to repair by appending "`~#~`" to each file, for example **`param.hist.~3~`**. The utility then writes the recovered information to the history database files, for example **`param.hist`**.

# File and Directory Structure Used by fix\_hist

PATROL creates and maintains a standard file and directory structure in which it stores the history database. The fix\_hist utility will not run if the proper file and directory structure do not exist.

## Directory Structure Used by the Fix History Utility

Table 12-8 lists the directories for fix\_hist.

**Table 12-8 Directories for fix\_hist**

| <b>Unix Directory</b>                              | <b>Definition</b>  |
|--|--|
| <b>Windows Directory</b>                           |  |
| <i>\$PATROL_HOME</i>                               | directory specified when the PATROL Agent is started   |
| <i>%PATROL_HOME%</i>                               |  |
| <i>\$PATROL_HOME/log</i>                           | log directory for the PATROL Agent   |
| <i>%PATROL_HOME%\log</i>                           |  |
| <i>\$PATROL_HOME/log/history</i>                   | top directory of the parameter history databases<br>contains top directories of each PATROL Agent  |
| <i>%PATROL_HOME%\log\history</i>                   |  |
| <i>\$PATROL_HOME/log/history/hostname</i>          | a PATROL Agent's top directory of the parameter history database. Created by the PATROL Agent using the official host name on which it is running                    |
| <i>%PATROL_HOME%\log\history\hostname</i>          |  |
| <i>\$PATROL_HOME/log/history/hostname/port_num</i> | the directory where a single PATROL Agent stores files when multiple PATROL Agents are running on the same host<br>created by the PATROL Agent using its port number |
| <i>%PATROL_HOME%\log\history\hostname\port_num</i> |  |

## Files Read by the Fix History Utility

Table 12-9 shows which files are read by the `fix_hist` utility.

**Table 12-9 Files Read by `fix_hist`**

| File Description  | Filename and Path for Unix  |
|-------------------|---|
|                   | Filename and Path for Windows   |
| Index file        | <code>\$PATROL_HOME/log/history/hostname/port_num/dir</code>          |
|                   | <code>%PATROL_HOME%\log\history\hostname\port_num\dir</code>          |
| History text file | <code>\$PATROL_HOME/log/history/hostname/port_num/annotate.dat</code> |
|                   | <code>%PATROL_HOME%\log\history\hostname\port_num\annotate.dat</code> |
| History data file | <code>\$PATROL_HOME/log/history/hostname/port_num/param.hist</code>   |
|                   | <code>%PATROL_HOME%\log\history\hostname\port_num\param.hist</code>   |

## Relocating Files

Moving these files will not prevent you from viewing them. The `dump_hist` utility can be used to view the old history data.

### To Reuse the Old History Data

Perform the following steps:

- Step 1** Create the directory `$PATROL_HOME/log/history/hostname/new_port` on Unix or `%PATROL_HOME%\log\history\hostname\new_port` on Windows.
- Step 2** Move the files in the **dir**, **annotate.dat**, and **param.hist** files to the new directory. (The files must be named **dir**, **annotate.dat**, and **param.hist**.)
- Step 3** To view the data, type the following command and press **Enter**:  
`dump_hist -p new_port`

## Before Running the fix\_hist Utility

The PATROL environment must be set up prior to running fix\_hist. You must set the PATROL\_HOME environment variable. Table 12-5, “PATROL\_HOME Environment Variable,” on page 12-13 describes how to define this variable for Unix and Windows.

## Running fix\_hist

Running the fix\_hist command could require shutting down the agent for a considerable amount of time. The length of the downtime is directly related to the size of **dir**, **annotate.dat**, and **param.hist**. If these files are very large or any downtime for the agent is undesirable, consider clearing unneeded history or moving the files instead of running fix\_hist. Some options for the fix\_hist command may reduce its run time. (See “Options” on page 12-25.)

---

### Warning

---

The fix\_hist utility requires disk space while it is running. If you have limited disk space on your computer, it is possible to run out of space. Running out of disk space can cause all PATROL files to be inaccessible or it can cause other applications to crash.

---

To run fix\_hist

- Step 1** Stop the PATROL Agent.
- Step 2** For Windows, set the variable called %PATROL\_HOME%. For Unix, run one of the following scripts from the PATROL installation directory:
- `./patrolrc.sh` (for Korn and Bourne shell)
  - `source .patrolrc` (for C shell)
- Step 3** At the command line, run fix\_hist and monitor the output.
- Step 4** Repeat Step 3 until no more errors are reported or until you determine that the history files can not be fixed.

## Return Values

The `fix_hist` utility exits with one of the following values:

- 0 `fix_hist` completed successfully
- 1 `fix_hist` was aborted because of an error

## Syntax

The `fix_hist` utility has the following format:

```
fix_hist    [-p port]
            [-host hostname]
            [-no_repair_hist]
            [-no_pack_hist_index]
            [-no_pack_ann]
            [-no_backup]
            [-repair_cycle number]
            [-v]
```

## Options

Table 12-10 lists and defines the options recognized by the `fix_hist` utility.

**Table 12-10 Options Recognized by the `fix_hist` Utility (Part 1 of 2)**

| Option                       | Definition   |
|------------------------------|--|
| <code>-p</code>              | This option specifies which PATROL Agent's history database is used. Combined with the <code>-host</code> option, <code>-p</code> specifies a unique PATROL Agent. The value of this option should be the port number to which the specified PATROL Agent is listening. The default value is 3181. |
| <code>-host</code>           | This option specifies which PATROL Agent's history database is used. By default, the name of the host on which <code>fix_hist</code> is started is used.   |
| <code>-no_repair_hist</code> | This option stops this command from repairing the history. This option is useful if you only want to pack the data and index files.  |

**Table 12-10 Options Recognized by the fix\_hist Utility (Part 2 of 2)**

| <b>Option</b>       | <b>Definition</b>  |
|---------------------|--|
| -no_pack_hist_index | This option specifies that the history index file is to be left unpacked.                      |
| -no_pack_ann        | This option specifies that the annotation file is to be left unpacked.                         |
| -no_backup          | This option prevents the history files from being backed up before the command starts.         |
| -repair_cycles      | This option specifies the maximum number of times to scan the history data while repairing it. |
| -v                  | This option displays version information.  |

## Examples

The following examples illustrate how to use the fix history utility.

### To Repair History For an Agent on a Specific Port

This command fixes history and packs the data and index files at port 5000.

```
fix_hist -p 5000
```

### To Scan the History Database a Maximum of Two Times

This command fixes history and packs the data and index files for host host1 at port 3181. It will scan the history file a maximum of two times.

```
fix_hist -host host1 -p 3181 -repair_cycle 2
```

### To Repair History Database Without Packing Index Files

This command fixes history for host host1 at port 3181. The history index file is not packed.

```
fix_hist -host host1 -p 3181 -no_pack_hist_index
```

### To Pack Index Files Without Repairing History Database

This command packs the files for host host1 at port 3181. The history files are not scanned and fixed.

```
fix_hist -host host1 -p 3181 -no_repair_hist
```



---

# PATROL Agent Security

This chapter gives background information about the methods of maintaining security for the PATROL Agent, lists default ownership and permissions for the PATROL Agent, and tells you how to change those ownerships and permissions. This chapter contains the following sections:

|  |       |
|--|-------|
| PATROL Agent Security Methods . . . . .                      | 13-2  |
| Using PATROL in an Environment with Firewalls . . . . .      | 13-3  |
| Using Access Control List . . . . .                          | 13-4  |
| Using Accounts . . . . .                                     | 13-4  |
| Default Ownership and Permissions for Directories . . . . .  | 13-4  |
| Default Ownership and Permissions for Files . . . . .        | 13-5  |
| Changing Ownership and Permissions on Unix . . . . .         | 13-6  |
| Single Sign-On for Windows 2000. . . . .                     | 13-8  |
| Establishing Secure Connections. . . . .                     | 13-8  |
| Supported Platforms, Operating Systems, and Components . . . | 13-8  |
| Compatibility . . . . .                                      | 13-9  |
| Security Model. . . . .                                      | 13-9  |
| Location of Files . . . . .                                  | 13-9  |
| Configuring PATROL Agent for Single Sign-On . . . . .        | 13-10 |
| Log Files . . . . .  | 13-10 |
| Configure Rights in Active Directory Domain Trees . . . . .  | 13-11 |
| Disabling Single Sign-On. . . . .                            | 13-11 |

# PATROL Agent Security Methods

Table 13-1 lists the methods available for maintaining security for the PATROL Agent and tells you where you can find instructions for changing security.

**Table 13-1 Methods for Maintaining Security for the PATROL Agent (Part 1 of 2)**

| <b>Method of Maintaining Security</b> | <b>How Security Is Established</b>   | <b>Task Reference</b>                                     |
|---------------------------------------|--|---|
| Security levels                       | Allows you to install one of five security level policies which secure the dataflow between the PATROL Agent, PATROL Consoles, and PATROL Console Server.  | See the <i>PATROL Security User Guide</i>                 |
| User account                          | The default account for commands executed by the agent is specified by the defaultAccount variable in the agent configuration file. The agent cannot run application discovery and parameters properly without a valid user name.  | "Changing Ownership and Permissions on Unix" on page 13-6 |
| User and host names                   | The Access Control List (ACL) is defined by an agent configuration variable. The ACL specifies which user names can be used with which machines when connecting with an agent. The ACL configuration variable is described in Chapter 5, "Managing Console Connections". | "Controlling Access to the Agent" on page 5-9             |

**Table 13-1 Methods for Maintaining Security for the PATROL Agent (Part 2 of 2)**

| Method of Maintaining Security               | How Security Is Established   | Task Reference  |
|--|---|---|
| Directory and file ownership and permissions | <p>The <b>\$PATROL_HOME/log</b> and <b>\$PATROL_HOME/config</b> directories are created when the PATROL Agent process is executed for the first time. At that time, the ownership and permissions of the PATROL Agent log and configuration directories are set.</p> <p>If the <b>\$PATROL_ADMIN</b> environment variable is set, it specifies the user that owns all of the newly created log and configuration files.</p> <p>If the <b>\$PATROL_ADMIN</b> environment variable is not set, the user <b>PATROL</b> owns all of the files by default.</p> | See "Changing Ownership and Permissions on Unix" on page 13-6 |
| PATROL Access Control                        | Access is controlled by configuration variables in <code>patrol.conf</code>   | <i>PATROL for Windows 2000 User Guide - Volume III</i>        |

## Using PATROL in an Environment with Firewalls

If your environment includes firewalls, you may have to modify the configuration of the firewall to accommodate PATROL. For information about installing and configuring PATROL in an environment with firewalls, see the *PATROL Installation Guide*.

# Using Access Control List

The Access Control List (ACL) controls which users are authorized to connect to an agent, in which modes and from which hosts. For information on how to setup an Access Control List, see “Controlling Access to the Agent” on page 5-9.

# Using Accounts

You can instruct the PATROL Agent to use separate accounts for individual applications and instances. For more information on how to specify which accounts are used for which commands, see Chapter 4, “Establishing Accounts and Ports.”

# Default Ownership and Permissions for Directories

The default ownership and permissions of the PATROL Agent log and configuration directories are set according to Table 13-2:

**Table 13-2 Directories for Ownership and Permissions of Agent Log**

| <b>Unix Directory<br/>Windows Directory</b>                            | <b>Owner</b>              | <b>Permissions</b>                    |
|--|---------------------------|---------------------------------------|
| <code>\$PATROL_HOME\log</code><br><code>%PATROL_HOME%\log</code>       | AgentSetup/defaultAccount | Unix = 0755<br>Windows = Full Control |
| <code>\$PATROL_HOME/bin</code><br><code>%PATROL_HOME%\bin</code>       | root                      | Unix = 6755<br>Windows = Full Control |
| <code>\$PATROL_HOME/config</code><br><code>%PATROL_HOME%\config</code> | AgentSetup/defaultAccount | Unix = 0755<br>Win = Full Control     |

# Default Ownership and Permissions for Files

Table 13-3 shows the default ownership and permissions of the log and configuration files:

**Table 13-3 Default Owner and Permissions of Log and Files**

| <b>Unix Directory<br/>Windows Directory</b> | <b>Owner</b>              | <b>Permissions</b>              |
|---|---------------------------|---------------------------------|
| config/config_ <i>host-port</i> .dat        | AgentSetup/defaultAccount | Unix = 0644<br>Windows = Change |
| config/config_ <i>host-port</i> .idx        | AgentSetup/defaultAccount | Unix = 0644<br>Windows = Change |
| log/PatrolAgent_ <i>host-port</i> .errs     | AgentSetup/defaultAccount | Unix = 0644<br>Windows = Change |
| log/history/ <i>host/port</i> /dir          | AgentSetup/defaultAccount | Unix = 0644<br>Windows = Change |
| log/history/ <i>host/port</i> /annotate.dat | AgentSetup/defaultAccount | Unix = 0644<br>Windows = Change |
| log/history/ <i>host/port</i> /param.hist   | AgentSetup/defaultAccount | Unix = 0644<br>Windows = Change |
| log/PEM_ <i>host-port</i> .log              | AgentSetup/defaultAccount | Unix = 0644<br>Windows = Change |

For information about changing the ownership and permissions for files, see “Changing Ownership and Permissions on Unix” on page 13-6.

## Changing Ownership and Permissions on Unix

---

**Summary:** Normally, all log and configuration files are owned by the user designated by the Agent Configuration variable `/agentSetup/defaultAccount`. If you require the files to be owned by another user, change the `PATROL_ADMIN` environment variable in the shell script `$PATROL_HOME/./patrolrc.sh` or `$PATROL_HOME/./patrolrc`. Then change the ownership of `$PATROL_HOME/log` and `$PATROL_HOME/config` directories and any files within those directories to the new user.

---

### Before You Begin

If the `$PATROL_HOME` directory resides in a local file system and you do not have permission to change the ownership of the log and configuration directories, you may have to logon as root to perform Step 2 and Step 3.

If the `$PATROL_HOME` directory resides in an NFS-mounted file system, you may have to logon as root in the NFS server machine where `$PATROL_HOME` physically resides in order to perform Step 2 and Step 3.

---

#### Note

---

The `/agentSetup/defaultAccount` and `OSdefaultAccount` variables can also be modified through configuration utilities. See “Modifying the Default Account Variable” on page 10-18.

---

### To Change Ownership and Permissions For Files and Directories

**Step 1** Perform the appropriate action:

- If you are using the `sh` or the `ksh`, modify the following line in `$PATROL_HOME/./patrolrc.sh`:  

```
PATROL_ADMIN=user;export PATROL_ADMIN
```
- If you are using the `csh`, modify the following line in `$PATROL_HOME/./patrolrc`:

```
setenv PATROL_ADMIN user
```

- Step 2** Change the ownership of *\$PATROL\_HOME/log* and *\$PATROL\_HOME/config* directories and any files within those directories by using the following command:

```
chown -R user $PATROL_HOME/log $PATROL_HOME/config
```

- Step 3** Change the group ownership by using the following command:

```
chgrp -R groupname $PATROL_HOME/log  
$PATROL_HOME/config
```

# Single Sign-On for Windows 2000

Single Sign-On (SSO) security enables a user to

- sign on once to a node (server or workstation) in a Windows 2000 domain and then freely access any SSO-supported PATROL components in the same Active Directory domain tree
- establish a secure connection through which PATROL components can communicate

SSO for PATROL for Windows 2000 uses Windows 2000's Security Support Providers Interface (SSPI) to implement Windows 2000 security services, which include Kerberos's mutual authentication.

## Establishing Secure Connections

When a PATROL component attempts to establish a connection to a PATROL Agent, the component passes a ticket to the agent. The agent verifies the component's credentials and either accepts or denies the connection request.

## Supported Platforms, Operating Systems, and Components

This security option is only available for homogeneous Windows 2000 environments. Table 13-4 lists all supported hardware platforms, operating systems, and PATROL components and their versions.

**Table 13-4 Single Sign-On Security Supported Platforms and Components**

| <b>Platform</b> | <b>Operating System</b>             | <b>PATROL Component</b> | <b>Version</b> |
|-----------------|-------------------------------------|-------------------------|----------------|
| Intel           | Windows 2000 Workstation and Server | PATROL QuickView        | 1.2.00         |
|                 |                                     | PATROL Agent            | 3.4.00         |

## Compatibility

SSPI-enabled components are only compatible with other SSPI-enabled components. Non-SSPI-enabled components cannot communicate with SSPI-enabled components.

### Consoles and Agents

At present, only PATROL QuickView and PATROL Agent for Windows 2000 are compatible. The following consoles and agents *cannot* communicate with SSPI-enabled consoles and agents:

- PATROL Console
  - for Unix
  - for Windows 2000
- PATROL Agent
  - for Unix
  - for Windows NT 4.0
  - for OpenVMS
  - for AS400

## Security Model

Windows 2000 Single Sign-On uses the Kerberos protocol, which is based upon a ticketing model. A Key Distribution Center (KDC), which in Windows 2000 is any domain controller, issues a Ticket Granting Ticket (TGT) for all users in its realm, which corresponds to its domain. When a user wants to use a service, it requests a Service Ticket (ST) from the KDC and presents it to the service.

## Location of Files

The PATROL Single Sign-On for Windows 2000 library, **esi\_ssapi.dll**, must reside in the **%PATROL\_HOME%\bin** directory.

## Configuring PATROL Agent for Single Sign-On

During the installation process of PATROL Single Sign-On for Windows 2000, the PATROL Agent is automatically configured to run Single Sign-On. You can manually disable or enable Single Sign-On configuration.

**Step 1** Specify the location of the library for Single Sign-On (**esi\_ssapi.dll**) using the PATROL Client Configuration Utility (**pccfg.exe**). Both **pccfg.exe** and the library are located in the **%PATROL\_HOME%\bin** directory.

**Step 2** Set the AgentSetup/security/ExtendedSecurityEnabled variable in the agent configuration to **yes**.

For more information on modifying variables in the agent configuration using configuration utilities, see Chapter 10, “Using wpconfig (Windows) to Configure the PATROL Agent.”

## Log Files

Single Sign-On does not have a separate log facility. When the PATROL Agent is in debug mode, the agent writes user log-on validation and rejection messages to the agent log file.

# Configure Rights in Active Directory Domain Trees

Windows 2000 provides security for environments that contain nested domains, which are also referred to as Forests. Administrators can create accounts and grant security privileges, also known as rights, at any domain level in the Active Directory Tree.

---

## Warning

---

Forests do not trust each other by default.

Objects in a forest do not automatically inherit rights from the top-level domain. If a domain does not directly branch off of a top-level domain (it's nested), you must explicitly grant rights to all objects owned by that domain.

---

## Disabling Single Sign-On

You can turn off the security provided by Single Sign-On by

- Setting the AgentSetup/security/ExtendedSecurityEnabled variable in the agent configuration to **no**.
- moving or deleting the configuration file from the **security** directory



---

# PATROL Agent and SNMP Concepts

This chapter provides information about how the PATROL Agent and PATROL SNMP Master Agent interact and what type of information the agents gather and send. It discusses the PATROL SNMP architecture and the roles that PATROL can play. This chapter contains the following sections:

|  |       |
|--|-------|
| PATROL SNMP Design . . . . .                               | 14-2  |
| Support . . . . .  | 14-2  |
| Architecture . . . . .                                     | 14-3  |
| Dependencies . . . . .                                     | 14-4  |
| Configuration Requirements . . . . .                       | 14-5  |
| Startup Process . . . . .                                  | 14-6  |
| SNMP Roles Available to the PATROL Agent . . . . .         | 14-8  |
| PATROL Agent as SNMP Manager . . . . .                     | 14-8  |
| Variables for Configuring the PATROL Agent . . . . .       | 14-8  |
| PSL Functions for Configuring the PATROL Agent . . . . .   | 14-10 |
| Two Methods of Sending SNMP Traps . . . . .                | 14-11 |
| Comparison of Methods . . . . .                            | 14-12 |
| Diagnostic Information for the PATROL SMUX based           |       |
| SNMP sub-agent . . . . .                                   | 14-13 |
| Creating Custom Read and Write Community Strings . . . . . | 14-13 |
| Modifying the snmpagt.cfg file . . . . .                   | 14-15 |

# PATROL SNMP Design

The PATROL SNMP architecture is based on the industry standard known as SMUX. One or more SNMP subagents can connect to a single SNMP Master Agent by using a TCP SMUX port (TCP port 199 by default).

The PATROL SNMP sub-agent can use the OS distributed SMUX SNMP Master Agent to communicate.

The BMC Patrol Agent SNMP Object Identifier is: enterprises.1031.1.1

The PATROL Agent has built-in functionality to allow it to act as an SNMP manager and as an SNMP agent functioning as a SMUX-based SNMP sub-agent. In order for Patrol SNMP to be functional, an SMUX compliant SNMP master agent or the BMC provided PATROL "snmpmagt" must be configured and running.

The SMUX based SNMP sub-agent can use any SMUX compliant SNMP Master agent as its master agent instead of the BMC provided PATROL "snmpmagt".

The PATROL SMUX based SNMP sub-agent is not compatible with SNMP master agents that are not built upon SMUX technology.

## Support

With PATROL SNMP support, you can integrate PATROL products into an existing SNMP Management system. To send traps to an SNMP manager, you can use

- PATROL's built-in trap send or trap notification based on PATROL events
- the PSL SNMP functions to send traps according to user-defined customizations

# Architecture

The PATROL SNMP architecture comprises an SNMP subagent that is part of the PATROL Agent and an SNMP Master Agent that is a separate, external process.

Figure 14-1 shows the relationships between the PATROL SNMP Master Agent, the PATROL Agent, PATROL SNMP subagents, SNMP subagents, and the SNMP manager. It also shows their default port numbers.

**Figure 14-1 PATROL SNMP Master Agent Interacts with Other Agents**

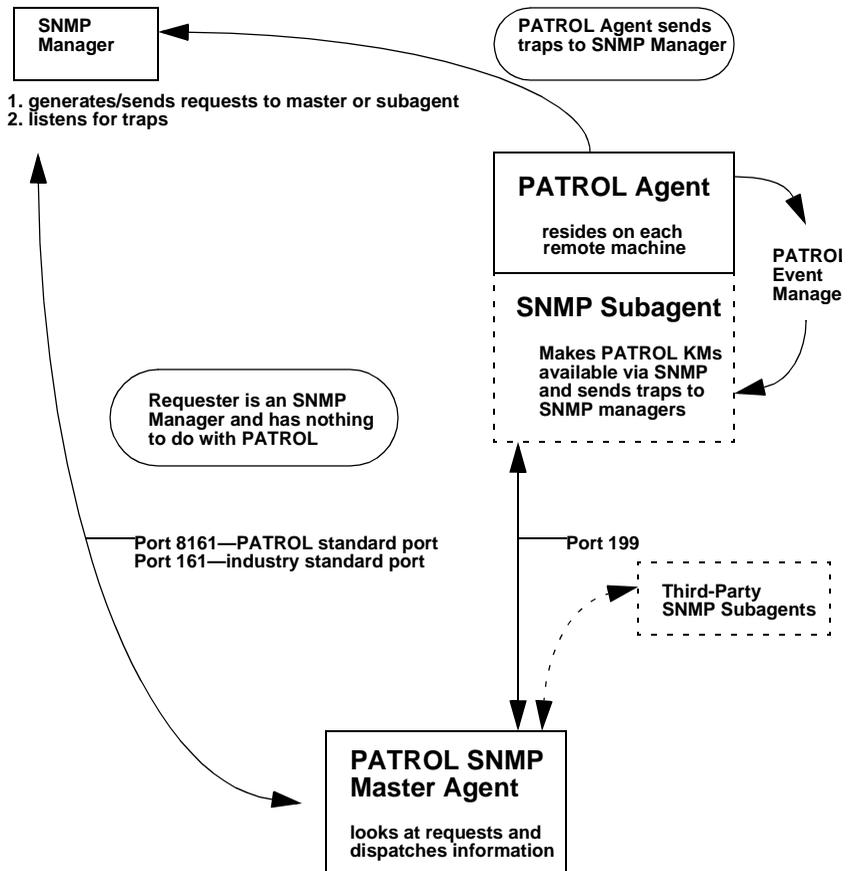


Table 14-1 describes the role that each component plays in the PATROL SNMP architecture.

---

**Note**

---

Port 161 is an industry standard, while port 8161 is a PATROL standard. By running on port 8161, you can avoid possible conflicts with operating system vendors who are already using port 161 for an SNMP agent.

---

**Table 14-1 PATROL SNMP Component Definitions**

| <b>Component</b>                                  | <b>SNMP Role</b>   |
|---|--|
| PATROL Agent                                      | The PATROL Agent generates traps.  |
| PATROL SNMP Master Agent                          | The PATROL SNMP Master Agent provides SNMP information to SNMP managers in the form of responses to SNMP queries.  |
| PATROL SNMP Subagent                              | An SNMP subagent responds to SNMP queries by going into internal data structures and sending information back. It can also generate or send PATROL traps directly to the SNMP Manager.       |
| PATROL SNMP Manager Functionality (PSL Functions) | The PATROL SNMP Manager receives traps and issues queries to an SNMP agent. The manager controls the agent by making SNMP requests of the agent and by setting variables in the agent's MIB. |

## Dependencies

The PATROL Agent communicates with both SNMP Managers and SNMP Agents. It communicates with the SNMP Managers through the SNMP Master Agent. For full functionality, the PATROL Master Agent must be running.

While you do not have to have the PATROL SNMP Master Agent and subagent running on the same computer, you do have to have an SNMP Agent that supports SMUX protocol because the subagent looks for a master agent. If the subagent does not find a master agent, it tries to start the PATROL SNMP Master Agent and tries to connect again.

By default, some operating systems do not use SMUX-compliant SNMP Master Agents.

The snmp sub-agent must be active if you want to use the following features:

- use PSL snmp\_trap\_send to send snmp traps
- send built-in automatic snmp traps (i.e. the snmp traps that get generated when events configured with SEND\_TRAP are generated)
- to make and respond to snmp requests

## Configuration Requirements

For PATROL to work properly with SNMP, the communication and configuration settings for the PATROL SNMP components must be set correctly. The following table lists the requirements, configuration data, and default settings for the PATROL SNMP components.

**Table 14-2 Configuration for PATROL Support of SNMP**

| Component                | Requirement   | Configuration Data   | Default Configuration Setting |
|--------------------------|---|--|-------------------------------|
| PATROL SNMP Subagent     | running   | /snmp/agent_auto_start   | yes                           |
|                          | communicates with PATROL SNMP Master Agent              | port 199   | 199—SMUX port                 |
| PATROL SNMP Master Agent | starts Master Agent                                     | SNMPStart parameter in the <b>platform.km</b> file               | active                        |
|                          | Master Agent SNMP listening port (standard port is 161) | port number  | 8161—SNMP listening port      |
|                          |   | \$PATROL_HOME/lib/snmpmagt.cfg (for UNIX)                        |                               |
|                          |   | %PATROL_HOME%\lib\snmpmagt.cfg (for Windows NT and Windows 2000) |                               |

**Table 14-2 Configuration for PATROL Support of SNMP**

| <b>Component</b> | <b>Requirement</b>                          | <b>Configuration Data</b>                          | <b>Default Configuration Setting</b>              |
|------------------|---|--|---|
| PATROL Agent     | supports SNMP                               | /snmp/support                                      | yes   |
|                  | list of SNMP Managers                       | /snmp/piVm_list                                    |   |
|                  | sends traps                                 | /AgentSetup/pemSNMP Support                        | yes   |
|                  | severity level of events that trigger traps | /AgentSetup/pemPFSnmp Nseverity                    | 1   |
|                  | read and write community names              | /snmp/agent_r_community<br>/snmp/agent_w_community | public (for PSL usage)<br>private (for PSL usage) |

## Startup Process

SNMP agent support is active only when both the PATROL SNMP Subagent and the PATROL SNMP Master Agent are communicating.

1. Assuming that the agent configuration variable /snmp/agent\_auto\_start is set to **yes**, the PATROL Agent attempts to start the PATROL SNMP Subagent. However, if the PATROL SNMP Master Agent is not running, the subagent fails and PATROL writes an error message to the PATROL error (.errs) log.

---

### Example

---

ID 102071: Fri MON DD HH:MM:SS CCYY: PATROL SNMP Subagent connection failed - Make sure SNMP Master Agent is running.

---

2. The *SNMPStart* parameter is defined within each **platform.km file**. The “out of box” default setting for this parameter is the active state. If active, *SNMPStart* checks to see if the PATROL SNMP Master Agent (*snmpmagt*) is running; if it is not, the parameter starts the master agent. *SNMPStart* then starts the PATROL SNMP Subagent and establishes a connection between the two.

When the PATROL SNMP Master Agent and the PATROL SNMP Subagent successfully connect, PATROL writes a message to the PATROL error log (.errs).

---

**Example**

---

ID 10209d: Fri MON DD HH:MM:SS CCYY: SNMP subagent is now connected.

---

# SNMP Roles Available to the PATROL Agent

Using a combination of agent configuration variables and PSL functions for SNMP, the PATROL Agent can act as an

- **SNMP Agent**—providing SNMP information to SNMP Managers in the form of responses to SNMP queries. See PATROL MIB.
- **SNMP Manager**—receiving traps and issuing queries to an SNMP Agent. The SNMP Manager controls the SNMP agent by making SNMP requests of the agent and by setting variables in the agent MIB.

## PATROL Agent as SNMP Manager

PATROL SNMP support can also be used to receive SNMP traps from other sources.

Converting a PATROL Agent into an SNMP trap handler requires PSL coding. For instructions on how to set up the agent as an SNMP manager, see “Configuring the PATROL Agent as an SNMP Manager” in Chapter 16, “SNMP Configuration and Implementation Using PSL.”

## Variables for Configuring the PATROL Agent

You configure the PATROL Agent to run with SNMP by changing the appropriate agent and configuration file variables. Table 14-3 shows each part of the process for configuring the PATROL Agent to run with SNMP and lists the variable or file that must be changed along with the section that contains information about it.

**Table 14-3 Configuring the PATROL Agent to Run with SNMP**

| <b>Task You Want to Perform</b>   | <b>Agent Configuration Variable</b>  | <b>Documentation</b>   |
|---|--|--|
| Set the port number and community name for the PATROL SNMP Master Agent.                                      | does not apply<br><br>This information is set in the <b>\$PATROL_HOME/lib/snmpmagt.cfg</b> . | The file comments describe the syntax.<br><br>snmpmagt.cfg is used to configure the PATROL "snmpmagt" (if used). |
| Turn on the SNMP support variable.  | <b>/snmp/agent_auto_start</b>  | "Start the PATROL SNMP Subagent Automatically" on page 15-9  |
| Bypass the master agent and send traps directly to a list of interested SNMP managers from the SNMP subagent. | <b>/snmp/trapMibTable</b>  | "Send Traps Directly to SNMP Managers" on page 15-14   |
|   | <b>/snmp/PiV1m_list</b>  | "Destination of SNMP PEM-based Traps" on page 15-20  |
| Configure events to send SNMP traps.  | standard or custom event catalog   | "Sending SNMP Traps Based on PATROL Events" on page 15-20  |
| Ignore IP Address of responses to SNMP "get" requests.  | /snmp/ignoreIPAddress  | "Ignoring Return IPAddresses" on page 15-14  |

### **SNMP Community Security**

The SNMP "COMMUNITY" is used for security reasons to allow SNMP manager to identify itself to an SNMP agent. The COMMUNITY clause is used within the Patrol SNMP Master Agent configuration file, \$PATROL\_HOME/lib/snmpmagt.cfg, to define external access to PatrolAgent SNMP sub-agent(s).

# PSL Functions for Configuring the PATROL Agent

Some of the PSL functions for SNMP configuration are briefly described in this section. Refer to the *PATROL Script Language Reference Manual* for detailed information about all PSL functions for SNMP.

## Sending SNMP Traps

As it sends traps, the PATROL Agent works in an SNMP agent role. Table 14-4 lists the function to use for the task you want to perform.

**Table 14-4 Functions for Sending Traps**

| Task You Want to Perform  | PSL Function to Use   |
|---|---|
| send any traps to any SNMP manager  | <code>snmp_trap_send()</code>   |
| send the trap <code>patrolTrapVlRaised</code> with <code>patrolTrapText.0</code> in a packet, to all entities registered in the <code>prVlmTable</code> | <code>snmp_trap_raise_std_trap("text")</code><br>The <code>prVm_list</code> must be defined to make this function work correctly. |

## Starting and Stopping the SNMP subagent

You can use PSL functions to stop, restart, and request the current state of the PATROL Agent. Table 14-5 lists the function to use for the task you want to perform.

**Table 14-5 Functions for Starting and Stopping the SNMP Agent**

| Task You Want to Perform                       | PSL Function to Use              |
|--|----------------------------------|
| request the current state of the SNMP subagent | <code>snmp_agent_config()</code> |
| start the SNMP subagent                        | <code>snmp_agent_start()</code>  |
| stop the SNMP subagent                         | <code>snmp_agent_stop()</code>   |

## Using PSL to Change the Registered SNMP Manager List

The list of registered SNMP Managers is contained in the `PivlmTable`. Table 14-6 lists the function to use for the task you want to perform.

**Table 14-6 Functions for Changing the Registered SNMP Manager List**

| <b>Task You Want to Perform</b>      | <b>PSL Function to Use</b>           |
|--------------------------------------|--------------------------------------|
| add an SNMP Manager to the list      | <code>snmp_trap_register_im()</code> |
| delete an SNMP manager from the list | <code>snmp_trap_register_im()</code> |
| print the list of SNMP Managers      | <code>snmp_trap_register_im()</code> |

## Two Methods of Sending SNMP Traps

The PATROL Agent can send SNMP traps to an SNMP management console by using any of the following methods:

- sending PATROL Events—sends a trap based on `TRAP_SEND` and `NO_TRAP` settings in event definitions

For details on how to configure the agent to send traps based on PATROL events, see Chapter 15, “SNMP Configuration and Implementation Using PEM.”

- running a PSL function—sends a trap based on user-defined rules coded in PSL

For details on how to configure the agent to send traps based on PSL functions, see Chapter 16, “SNMP Configuration and Implementation Using PSL.”

## Comparison of Methods

Table 14-7 compares the differences between the SNMP trap sending methods.

**Table 14-7 Comparing Methods for Sending Traps**

| <b>SNMP Trap Features</b>                         | <b>PEM Traps</b>                                     | <b>PSL Traps</b>                     |
|---|--|--------------------------------------|
| requires configuration of out-of-box installation | yes  | no                                   |
| any trap format possible                          | no   | yes                                  |
| enterprise OID can be changed                     | no   | yes                                  |
| different OID possible for each KM class          | no   | yes                                  |
| trap message can be configured or changed         | no   | no                                   |
| number of different trap formats possible         | 2  | 2                                    |
| methods of controlling format of these traps      | event catalog settings and agent configuration       | PSL coding, almost unlimited options |
| situations causing trap sending                   | generation of an event in the associated event class | any method of PSL execution          |

## Diagnostic Information for the PATROL SMUX based SNMP sub-agent

Make sure the PatrolAgent snmp sub-agent is running by entering the following at the system output window:

```
OS> %PSL print(snmp_agent_config());
```

If it has not started, enter the following:

```
OS> %PSL print(snmp_agent_start());
```

If 'ERR' is returned, the master agent communication is not working.

## Creating Custom Read and Write Community Strings

The following steps allow you to customize READ and WRITE community strings and change from "public" and "private" to other names.

1. If a community string is set different from "public" or "private", then you need to modify some Patrol Agent variables to coincide with your changes. From a Patrol Developer Console, use WPCONFIG or XPCONFIG to set the following variables:

- `/snmp/agent_r_community "<your_public_name>"`
- `/snmp/agent_w_community "<your_private_name>"`
- `/snmp/default_r_community "<your_public_name>"`
- `/snmp/default_w_community "<your_private_name>"`

You do not need to restart the PATROL agent to apply these changes

2. Use the Patrol Developer console to enter the following PSL command to determine if the SNMP sub-agent is active:

```
%PSL print(snmp_agent_config());
```

If the SNMP sub-agent is active, this command returns the following message: "SNMP Support is Active."

If it returns a message saying "SNMP Support is not Active", use the following command to start it.

```
print(snmp_agent_start());
```

If "OK" is returned, all is well. If ERR is returned, the master agent communication is not working. The most likely reason for this is that the SNMP Master Agent is not running. For NT, use Task Manager to search for the snmpmagt.exe process. For Unix platforms, use `ps -ef | grep snmpmagt`. If the snmp master agent is not started, start it.

The following section provides details for starting the SNMP Master Agent.

3. Once you have established that the SNMP sub-agent is running, test the SNMP communications. If you can not communicate, check the `$PATROL_HOME/lib/snmpmagt.cfg` file for a valid MANAGER and COMMUNITY name. This file allows you to define which MANAGERS receive which SNMP traps.

You can also define COMMUNITY, which allows you to configure the agent to only accept requests from certain MANAGERS with specified community strings.

---

**Note**

---

You can define COMMUNITYs by placing “ (quotes) around the strings in the following section.

---

```
COMMUNITY "<your_public_name>"
ALLOW ALL OPERATIONS
USE NO ENCRYPTION
```

Next, stop and restart the SNMP master agent process (snmpmagt.exe). On Windows, use the Task Manager to End Task. On Unix platforms, login as root and kill the SNMP master agent process (snmpmagt).

Then use a Patrol Developer console, select the PatrolAgent host icon, and use the following menu command to start the SNMP master agent:

- Windows systems: MB3=>KM Commands->SNMP Reconfigure.
- Unix systems: MB3=>Start SNMP Master Agent

This stops and restarts the SNMP master agent. Allow a couple of minutes and then verify that the SNMP master agent is running:

- Windows systems: use Task manager to look for snmpmagt.exe.
- Unix systems: use `ps -ef | grep snmpmagt`.

If this fails, simply stop and restart the PatrolAgent.

4. For Windows systems, make sure that the Microsoft SNMP service is configured correctly. Do the same for SNMP Managers on other platforms.

To verify that the Microsoft SNMP service is configured correctly, perform the following steps:

» Select **Control Panel->Networking->Services->Properties**

Verify that the traps tab has your Community Name `<public_name>` and the Security Tab has the same Accepted Community Names set to `<public_name>`.

You must restart the MS SNMP service.

## Modifying the snmpagt.cfg file

The SNMP "COMMUNITY" is used for security reasons to allow SNMP manager to identify itself to an SNMP agent. The COMMUNITY clause is used within the Patrol SNMP Master agent configuration file, `$PATROL_HOME/lib/snmpmagt.cfg`, to define external access to PatrolAgent SNMP sub-agent(s).

Within Patrol, the default community string is "public"; however, you may need to change the community. The default COMMUNITY clause within the SNMP Master Agent configuration file is:

```
COMMUNITY public
ALLOW ALL OPERATIONS
USE NO ENCRYPTION
```

Within the Patrol master agent configuration, the default community is "public"; however, you may need to change the community in order to restrict SNMP access to the PatrolAgent SNMP.

The following formats are acceptable for entering COMMUNITY string in \$PATROL\_HOME/lib/snmpmagt.cfg:

- A series of ASCII characters (both numeric and non-numeric characters are allowed) beginning with a non-numeric character with no double quotes. For example:

"public" is entered as: COMMUNITY public.

- A series of ASCII characters (both numeric and non-numeric characters are allowed) beginning with either a numeric or non-numeric character surrounded by quotes. For example:

"public" would be entered as: COMMUNITY "public"

"0public" would be entered as: COMMUNITY "0public"

- A series of hex digits representing ASCII representation of characters preceded by 0x. For example:

"public" would be entered as: COMMUNITY 0x7075626C6963.

The following error message is generated when the COMMUNITY that is entered violates the rules described above:

```
<$PATROL_HOME>/lib/snmpmagt.cfg, line 1 near "@": syntax error  
./snmpmagt: error processing configuration
```

For a complete definition of SNMP Master Agent configuration syntax, see the *Patrol SNMP Toolkit General Porting Guide*.

To maintain consistency when changing the COMMUNITY in the snmpmagt.cfg file, also update the following agent configuration variables:

- /snmp/agent\_r\_community "<your\_public\_name>"

- /snmp/agent\_w\_community "<your\_private\_name>"
- /snmp/default\_r\_community "<your\_public\_name>"
- /snmp/default\_w\_community "<your\_private\_name>"

## Basic Agent Configuration File

The agent configuration file controls several aspects of agent operation. Those aspects peculiar to SNMPv2 are covered in a later section. The ones with generic applicability are covered here, and fall into the following categories:

- initial values for sysContact and sysLocation
- community-based access control
- community-based naming
- trap destinations
- choice of transport protocols
- sub-agent access control
- sub-agent protocol parameters
- relationship of subagents to community-based naming

The remainder of this section, through descriptions of the configuration file grammar and examples, shows how you can use the agent configuration file to meet your needs. The MultiMIB Agent and Optima OptiMaster releases 1.6 and later do accept the more limited configuration file used by toolkit releases 1.5d and earlier.

For background information, see RFC 1157 for an extensive discussion of how community-based SNMP access control works. A number of concepts from SNMPv2 are used here, even when SNMPv2 protocol is not in use. For background information, see RFC 1445, Administrative Model for version 2 of the *Simple Network Management Protocol (SNMPv2)*.

## Initial Values for `sysContact` and `sysLocation`

The `INITIAL` directive allows you to specify the initial values for the `sysContact` and `sysLocation` variables from MIB-II. The grammar for an initialization directive is as follows:

```
INITIAL <varName> <string>

<varName> ::= sysLocation |
              sysContact
```

If the string contains spaces, line breaks, tabs, etc., it must be enclosed in quotes. You may also specify the value in hexadecimal notation. The configuration file processing routines handle mapping `<CR>` and `<LF>` characters into NVT (Network Virtual Terminal) strings. See RFC 854 for a specification of the NVT conventions.

Note that these directives have an effect only when the agent is started with a non-existent non-volatile memory file. Changing the values in the configuration file and forcing the file to be re-read using a HUP signal does not change the values of the MIB variables in order to avoid confusing any managers which may have re-configured those variables.

The following example illustrates the use of the `INITIAL` directive in a configuration file.

```
INITIAL sysLocation "Computer Room A
1190 Saratoga Avenue
Suitet 130
San Jose, CA 95129-3343
USA"
```

```
INITIAL sysContact "Patrick Rockecharlie
Email: <patrick_rockecharlie@peer.com>
Voice: +1 408 556-0720
Fax: +1 408 556-0735"
```

The actual values you choose will be specific to each system. If no INITIAL directive is present for a variable, the default value chosen by the agent is the empty string unless you customize the startup code in module `system.c` in the following directory:  
`agent/snmpsm/src/wrk`

## Identifying Transport Mappings

The master agent may use multiple interfaces for SNMP and SMUX traffic. Each interface the master agent uses for listening for SNMP packets or SMUX connections requires one TRANSPORT definition in the configuration file. If no transport definitions are present, the defaults are used. If any SNMP transport mappings are present in the configuration, the default is not assumed.

Each transport mapping describes an interface over which the master agent listens for SMUX connections from sub-agents or SNMP requests from managers.

Sub-agents do not derive SMUX transport mappings from the agent configuration file. When a sub-agent calls `mgmt_init_env()` to initialize its management environment, if the first parameter is NULL, the SMUX port number to send to is derived from the TCP/IP stack's `/etc/services` file, or equivalent; if no SMUX port number is specified in this file, the port number specified in the file `include/ame/portnums.h` is used.

The syntax for the TRANSPORT clause in the agent configuration file is shown below. When `appl_protocol` is SNMP, `transport_protocol` defaults to UDP. If `appl_protocol` is SMUX, `transport_protocol` defaults to TCP. The IP address in `faddr` defaults to that of the master agent's host. The IP port number defaults as described above.

The syntax for the TRANSPORT clause in the agent configuration file looks like this:

```
TRANSPORT <name> SNMP
```

```
[OVER UDP SOCKET]  
[AT <addr>]
```

```
TRANSPORT <name> SMUX  
[OVER TCP SOCKET]  
[AT <addr>]
```

```
addr ::= <ip-kind> | <rfc1449addr> | <full-ip>  
ip-kind ::= <hostid> |  
<hostid> <portid> |  
<portid>
```

```
hostid ::= <hostname> | <ip>
```

```
portid ::= PORT | : ] <#>
```

```
full-ip ::= <ip>:<#>  
ip ::= <#>.<#>.<#>.<#>
```

```
rfc1449addr ::= <ip>/<#>
```

Note that there are many ways to enter an IP address.

The following example configures a master agent to accept SMUX connections at both the standard port and at a non-standard port. It also configures the agent to accept SNMP traffic only at a non-standard port.

```
TRANSPORT      extraordinary SNMP  
                OVER UDP SOCKET  
                AT PORT 11161
```

```
TRANSPORT      ordinary SMUX  
                OVER TCP SOCKET  
                AT PORT 199
```

TRANSPORT      special SMUX  
OVER TCP SOCKET  
AT PORT 11199

The maximum number of concurrent SNMP and SMUX transports will be limited by your target system's limits on the number of open sockets or file descriptors per process.

## Using Community Strings

Community strings serve many functions in SNMP. Their most common use is to provide a weak form of authentication. In this capacity, they are part of the access control framework for SNMP. They are also used to help name pieces of information in the network. The following example of how communities can be used may be helpful:

- at IP address 192.146.153.65 at port 161, using community string "public", the value of {sysContact 0} is "Fred", and only read access is permitted.
- at IP address 192.146.153.65 at port 161, using community string "RenMinRiBao", the value of {sysContact 0} is "Fred", but both read and write access are permitted.
- at IP address 192.146.153.65 at port 161, using community string "subsystem1", the value of {sysContact 0} is "Joe", and only read access is permitted.

In this example, two of the community strings provide access to the same information, but with different access control policies. This third community string provides access to another set of information. Note that the manager requires external knowledge to determine whether the first two refer to the same information or to two different pieces of information that happen to have the same value.

A community string may be thought of as a key into a database table with the following information:

- Spatial Semantics: *which* set of information (where a system may have multiple instances of the same information classes)

- Temporal Semantics: *when* was the information retrieved or when will changes take effect (there may be caching, changes may require a reboot to take effect, etc.)
- Access Control Policy: *who* is allowed to use this information? What operations are allowed?

By now it should be clear that the community string is seriously overloaded. To support all the semantics carried by community strings, there are several configuration file entries.

COMMUNITY entries determine which community strings are valid. If no COMMUNITY entries are present, any community will be accepted by the agent. For each community, the set of supported operations is specified by the ALLOW clause. The MEMBERS clause in a community entry restricts the set of managers allowed to use that community. If the MEMBERS clause is absent from a COMMUNITY entry, any manager may use that community.

```
<communityDefinition> ::= COMMUNITY <communityName>
    ALLOW <op> [ , <op> ] * [ OPERATIONS ]
    AS ENTITY <entityName> ]
    MEMBERS <addr> [ , <addr> ] ]
```

```
<communityName> ::= <name>
    <op> ::= ALL | GET | SET | TRAP |
    RESPONSE | GET-NEXT | GET-BULK |
    INFORM | NOTIFY
```

The keyword ALL serves as a shorthand for listing the possibilities.

The optional AS ENTITY clause provides any naming semantics associated with this community. An entity definition associates spatial/temporal properties with a label. The spatial/temporal entity is defined using the following grammar:

```
<entityDefinition> ::= ENTITY <EntityName>
    DESCRIPTION <String>
    WITH TIME DOMAIN <oid> ]
```

<entityName> ::= <name>

The following example shows how a system might be configured to provide read-only and read-write access both to its standards resources as well as a special subsystem.

```
ENTITY SpecialSubsystem
    DESCRIPTION "The ultra-enhanced subsystem"
```

```
COMMUNITY public
    ALLOW GET OPERATIONS
```

```
COMMUNITY public1
    ALLOW GET OPERATIONS
    AS ENTITY SpecialSubsystem
```

```
COMMUNITY "YingXio'ng"
    ALLOW ALL OPERATIONS
```

```
COMMUNITY "Jia-oTo-ng"
    ALLOW ALL OPERATIONS
    AS ENTITY SpecialSubsystem
    MEMBERS bigboss.megalith.com, 192.146.153.65
```

Thus, to access information from the "ultra-enhanced subsystem", a manager would have to use either community "public1" or "Jia-oTo-ng". Using community "public" or "YingXio'ng" would access the default spatial-temporal entity. Furthermore, use of the community "Jia-oTo-ng" is restricted to operations with a source address from system "bigboss.megalith.com" or 192.146.153.65; operations with any other source address would not be accepted with this community.

Note the close relationship between the community-to-entity mapping and the sub-agent-to-entity association described in the section on controlling sub-agent communications, and how to use it to extend SNMP naming by community string.

To avoid unintended access, be sure the ENTITY, COMMUNITY and ALLOW/DENY SUB\_AGENT specifications are complete; that is, taken together, the definitions in the configuration file should cover all aspects of the configuration.

## Controlling Trap Destinations

MANAGER entries in the configuration file define SNMP trap destinations. Destinations specified in this manner are independent of destinations specified by means of the SNMPv2 party MIB.

```
<trapDestination> ::= MANAGER <addr>
                        [ON TRANSPORT <transportName>]
                        [SEND [ALL | NO] TRAPS
                        [TO PORT <#> ]
                        [WITH COMMUNITY <String>]]
```

See above for a description of the meaning of `transportName`. If it is not supplied, whatever is being used for primary SNMP traffic will be used.

If the `PORT` clause is not supplied, `snmp-trap` from your systems `/etc/services` or whatever you have built into the agent by editing `include/ame/portnums.h` will be used.

If the `COMMUNITY` clause is absent, the community string "public" will appear in the traps that are sent.

The following example requests that all traps be sent to manager "fred" at both port 162 and 11162, from the non-standard snmp port defined above, using a community string of "deep dark secret".

```
MANAGER      fred ON TRANSPORT extraordinary
              SEND ALL TRAPS TO PORT 162
              WITH COMMUNITY "deep dark secret"
              SEND ALL TRAPS TO PORT 11162
              WITH COMMUNITY "deep dark secret"
```

The number of trap destinations is limited only by available memory.

## Reloading the Configuration File

Under Unix, the agent may be told to re-read its configuration file by giving the agent process a HUP signal. Under Windows, the master agent's window has a menu item to reload the file. This allows you to change access restrictions without stopping/restarting the agent. Error messages go to the agent's `stderr`, which is not necessarily the same as that of the Unix process issuing the HUP signal. This action causes the agent to issue a warm start trap, if traps are enabled in the agent configuration file.



---

# SNMP Configuration and Implementation Using PEM

This chapter discusses how to use PATROL's built-in SNMP support to send traps based on PATROL events. This chapter contains the following sections:

|   |       |
|---|-------|
| Before You Begin Configuring PATROL with SNMP . . . . .           | 15-3  |
| Process for Configuring the PATROL Agent with SNMP . . . . .      | 15-3  |
| When Configuration Changes Take Effect . . . . .                  | 15-4  |
| Configuring the PATROL SNMP Master Agent . . . . .                | 15-4  |
| Starting The PATROL SNMP Master Agent . . . . .                   | 15-4  |
| Specify the Working Directory . . . . .                           | 15-8  |
| Starting the PATROL SNMP Master Agent . . . . .                   | 15-8  |
| Configuring the SNMP Subagent and the PATROL Agent . . . . .      | 15-9  |
| Start the PATROL SNMP Subagent Automatically . . . . .            | 15-9  |
| Prevent the PATROL SNMP Master Agent on Unix from                 |       |
| Starting Automatically . . . . .                                  | 15-11 |
| Specify the Read Community Name . . . . .                         | 15-11 |
| Specify the Write Community Name . . . . .                        | 15-12 |
| Specify the Port Number for Sending and Receiving Traps . . . . . | 15-12 |
| Specify the MIB-II Information . . . . .                          | 15-13 |
| Send Traps through PATROL SNMP Master Agent . . . . .             | 15-13 |
| Send Traps Directly to SNMP Managers . . . . .                    | 15-14 |
| Ignoring Return IPAddresses . . . . .                             | 15-14 |
| Configuring the PATROL Event Manager to Send SNMP Traps . . . . . | 15-15 |
| List of Standard Event Classes . . . . .                          | 15-16 |

|   |       |
|---|-------|
| Sending SNMP Traps Based on PATROL Events . . . . .         | 15-20 |
| Destination of SNMP PEM-based Traps . . . . .               | 15-20 |
| SNMP Support Based on PATROL Event Manager. . . . .         | 15-21 |
| Filter Traps Based on PATROL Event Severity Level . . . . . | 15-22 |
| Specify Application That Issued the Trap . . . . .          | 15-23 |
| Specify the Node Name Where the Trap Originated . . . . .   | 15-23 |
| Filter Traps Based on PATROL Event ID. . . . .              | 15-24 |
| Filter Traps Based on PATROL Event Class . . . . .          | 15-24 |
| Filter Events Based on PATROL Event Description . . . . .   | 15-25 |
| Specify a Time Period to Send Traps Based on                |       |
| PATROL Events . . . . .                                     | 15-25 |
| Filter Traps Based on PATROL Event Type. . . . .            | 15-27 |
| Filter Traps Based on PATROL Event Status. . . . .          | 15-28 |
| Set SNMP Trap Format . . . . .                              | 15-29 |
| Disabling SNMP Trap Support for PATROL Events . . . . .     | 15-31 |

# Before You Begin Configuring PATROL with SNMP

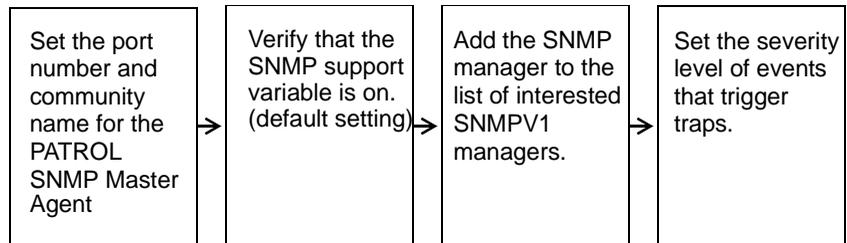
Before you begin configuring the PATROL Agent to run with SNMP, make sure:

- You have read and understood Chapter 3, “Background About Configuring the PATROL Agent.”
- You know the port number and community name to use for the PATROL SNMP Master Agent.
- You know the names of the SNMP managers to add to the interested manager list.
- You know the level of events for which you want to send traps.

## Process for Configuring the PATROL Agent with SNMP

You configure the PATROL Agent to run with SNMP by setting the appropriate variables. Figure 15-1 illustrates the process for configuring the PATROL Agent to run with SNMP.

**Figure 15-1 Configuring the PATROL Agent with SNMP**



## When Configuration Changes Take Effect

Changes made to the PATROL SNMP Master Agent configuration take effect after the agent restarts. These changes are persistent; they remain in effect regardless of how many times the PATROL SNMP Master Agent is shut down and restarted.

## Configuring the PATROL SNMP Master Agent

You can specify the working directory for the PATROL SNMP Master Agent and the start line (command string) that starts the PATROL SNMP Master Agent.

## Starting The PATROL SNMP Master Agent

The `/snmp/masterAgentStartLine` configuration variable controls how the PATROL SNMP Master Agent starts.

|                                |   |
|--------------------------------|---|
| <b>Format and Type of Data</b> | text string, not applicable   |
| <b>Default Value</b>           | <code>./snmpmagt \$PATROL_HOME/snmpmagt.cfg</code><br><code>NOV&amp;</code> |
| <b>Minimum and Maximum</b>     | not applicable  |
| <b>Dependencies</b>            | none  |
| <b>Recommendation</b>          | This variable is for Unix platforms only.                                   |

## Name of Executable

The `/snmp/masterAgentName` configuration variable specifies the name of the PATROL SNMP Master Agent executable file. This variable is used to build the start line.

|                                |   |
|--------------------------------|---|
| <b>Format and Type of Data</b> | text string, SNMP configuration file name |
| <b>Default Value</b>           | snmpmagt                                  |
| <b>Minimum and Maximum</b>     | not applicable                            |
| <b>Dependencies</b>            | none                                      |
| <b>Recommendation</b>          | none                                      |

## Name of Directory

The `/snmp/masterAgentDir` configuration variable specifies the name of the directory that contains the PATROL SNMP Master Agent executable file. This variable is used to build the start line.

|                                |                             |
|--------------------------------|-----------------------------|
| <b>Format and Type of Data</b> | text string, directory path |
| <b>Default Value</b>           | bin                         |
| <b>Minimum and Maximum</b>     | not applicable              |
| <b>Dependencies</b>            | none                        |
| <b>Recommendation</b>          | none                        |

## Name of Configuration File

The `/snmp/masterAgentConfigName` configuration variable specifies the name of the PATROL SNMP Master Agent configuration file. This variable is used to build the start line.

|                                |                                      |
|--------------------------------|--------------------------------------|
| <b>Format and Type of Data</b> | text string, configuration file name |
| <b>Default Value</b>           | snmpmagt.cfg                         |
| <b>Minimum and Maximum</b>     | not applicable                       |
| <b>Dependencies</b>            | none                                 |
| <b>Recommendation</b>          | none                                 |

## Name of Configuration File Directory

The `/snmp/masterAgentConfigDir` configuration variable specifies the name of the directory that contains the PATROL SNMP Master Agent configuration file. This variable is used to build the start line.

|                                |                             |
|--------------------------------|-----------------------------|
| <b>Format and Type of Data</b> | text string, directory path |
| <b>Default Value</b>           | lib                         |
| <b>Minimum and Maximum</b>     | not applicable              |
| <b>Dependencies</b>            | none                        |
| <b>Recommendation</b>          | none                        |

## Name of Nonvolatile Information File

The `/snmp/masterAgentParamName` configuration variable specifies the name of the PATROL SNMP Master Agent nonvolatile information file. This file is a flag file required for starting the master agent. It should not be moved or edited. This variable is used to build the start line.

|                                |                                |
|--------------------------------|--------------------------------|
| <b>Format and Type of Data</b> | text string, file name         |
| <b>Default Value</b>           | NOV                            |
| <b>Minimum and Maximum</b>     | not applicable                 |
| <b>Dependencies</b>            | none                           |
| <b>Recommendation</b>          | Do not move or edit this file. |

## Name of Nonvolatile Information File Directory

The `/snmp/masterAgentParamDir` configuration variable specifies the name of the directory that contains the PATROL SNMP Master Agent nonvolatile information file. This variable is used to build the start line.

|                                |                             |
|--------------------------------|-----------------------------|
| <b>Format and Type of Data</b> | text string, directory path |
| <b>Default Value</b>           | log                         |
| <b>Minimum and Maximum</b>     | not applicable              |
| <b>Dependencies</b>            | none                        |
| <b>Recommendation</b>          | none                        |

## Specify the Working Directory

The `/snmp/masterAgentWorkingDir` configuration variable specifies the working directory for the PATROL SNMP Master Agent on Unix. The working directory contains the PATROL SNMP Master Agent executable file.

This directory must contain the start line for the PATROL SNMP Master Agent on Unix.

|                                |                                |
|--------------------------------|--------------------------------|
| <b>Format and Type of Data</b> | text string, directory path    |
| <b>Default Value</b>           | <code>\$PATROL_HOME/bin</code> |
| <b>Minimum and Maximum</b>     | not applicable                 |
| <b>Dependencies</b>            | none                           |
| <b>Recommendation</b>          | none                           |

## Starting the PATROL SNMP Master Agent

The PATROL SNMP Master Agent is launched separately from the PATROL Agent. A command with a built-in computer class KM checks the `SNMPStart` parameter of the OS knowledge module. If the parameter is set to `yes`, PATROL automatically starts the PATROL SNMP Master Agent.

# Configuring the SNMP Subagent and the PATROL Agent

The PATROL SNMP Subagent is built into the PATROL Agent. Agent configuration variables control the configuration of both components.

## Start the PATROL SNMP Subagent Automatically

The `/snmp/agent_auto_start` configuration variable specifies whether the PATROL SNMP Subagent attempts to connect to the SNMP Master Agent when the PATROL Agent starts.

---

### Note

---

If this variable is set to **No** or if the attempt to start the subagent fails, the PATROL platform knowledge module's `SNMPStart` parameter starts the Master Agent and establishes a connection between the subagent and the master agent.

---

|                                |  |
|--------------------------------|--|
| <b>Format and Type of Data</b> | boolean, yes or no   |
| <b>Default Value</b>           | yes  |
| <b>Minimum and Maximum</b>     | not applicable   |
| <b>Dependencies</b>            | none   |
| <b>Recommendation</b>          | <p>Because the PATROL SNMP Master Agent is started by the <code>SNMPStart</code> parameter of the appropriate OS KM, when the agent starts, the master agent may not be running and the PATROL SNMP Subagent's attempt to connect will fail. The two conditions in which the master agent might be running are</p> <ul style="list-style-type: none"><li>• another PATROL Agent running on a different port started the PATROL SNMP Master Agent</li><li>• your environment uses a third-party SNMP Master Agent that runs independently of PATROL</li></ul> |

## **Error Message Generated**

When the subagent fails to connect to the master agent, PATROL writes the following message to the PATROL Agent error log.

```
ID 102071: Sat May 27 16:26:52 CCYY: Patrol SNMP subagent connection failed - Make sure SNMP Master Agent is running.
```

# Prevent the PATROL SNMP Master Agent on Unix from Starting Automatically

The `/snmp/masteragent_auto_start` configuration variable specifies whether the `SNMPStart` parameter starts the PATROL SNMP Master Agent when the platform KM is loaded.

---

## Note

---

This variable only affects the PATROL SNMP Master Agent. You must create the variable; it is not created during the installation of the PATROL Agent.

---

|                                |                    |
|--------------------------------|--------------------|
| <b>Format and Type of Data</b> | boolean, yes or no |
| <b>Default Value</b>           | yes                |
| <b>Minimum and Maximum</b>     | not applicable     |
| <b>Dependencies</b>            | none               |
| <b>Recommendation</b>          | none               |

## Specify the Read Community Name

The `/snmp/agent_r_community` configuration variable specifies the read community name for PATROL SNMP Master Agent operations.

|                                |                               |
|--------------------------------|-------------------------------|
| <b>Format and Type of Data</b> | text string, community access |
| <b>Default Value</b>           | public                        |
| <b>Minimum and Maximum</b>     | not applicable                |
| <b>Dependencies</b>            | none                          |
| <b>Recommendation</b>          | none                          |

## Specify the Write Community Name

The `/snmp/agent_w_community` configuration variable specifies the write community name for PATROL SNMP Master Agent operations.

|                                |                               |
|--------------------------------|-------------------------------|
| <b>Format and Type of Data</b> | text string, community access |
| <b>Default Value</b>           | private                       |
| <b>Minimum and Maximum</b>     | not applicable                |
| <b>Dependencies</b>            | none                          |
| <b>Recommendation</b>          | none                          |

## Specify the Port Number for Sending and Receiving Traps

The `/snmp/trap_port` configuration variable specifies the default listening port for PATROL SNMP Master Agent operations.

|                                |                      |
|--------------------------------|----------------------|
| <b>Format and Type of Data</b> | numeric, port number |
| <b>Default Value</b>           | 162                  |
| <b>Minimum and Maximum</b>     | not applicable       |
| <b>Dependencies</b>            | none                 |
| <b>Recommendation</b>          | none                 |

## Specify the MIB-II Information

MIB-II information is supported by the master agent.

## Send Traps through PATROL SNMP Master Agent

The `/snmp/trapConfTable` configuration variable specifies whether or not the PATROL SNMP Subagent sends traps to the SNMP managers through the PATROL SNMP Master Agent. A list of managers is set in the PATROL SNMP Master Agent configuration file and is not created through the PSL function, `snmp_trap_register_im()`.

|                                |  |
|--------------------------------|--|
| <b>Format and Type of Data</b> | boolean, yes or no   |
| <b>Default Value</b>           | no   |
| <b>Minimum and Maximum</b>     | not applicable   |
| <b>Dependencies</b>            | <p>This variable and the <code>snmp/trapMibTable</code> variable <i>are not mutually exclusive</i>. If both variables are set to <b>yes</b>, every time an activity satisfies the send-trap criteria, the PATROL Agent will send two traps, one to the SNMP managers listed in the <code>snmpmagt.cfg</code> file and one to the SNMP managers listed in <code>/snmp/piV1m_list</code>.</p> <p>If an SNMP Manager is in the <code>snmpmagt.cfg</code> file and the <code>/snmp/piV1m_list</code> agent configuration variable, the manager will receive the same trap twice.</p> |
| <b>Recommendation</b>          | none   |

## Send Traps Directly to SNMP Managers

The `/snmp/trapMibTable` configuration variable specifies whether or not the PATROL SNMP Subagent bypasses the PATROL SNMP Master Agent when it sends traps. When this variable equals **yes**, the subagent issues SNMP traps directly to the SNMP managers specified in the `/snmp/piV1m_list` agent configuration variable.

|                                |  |
|--------------------------------|--|
| <b>Format and Type of Data</b> | boolean, yes or no   |
| <b>Default Value</b>           | yes  |
| <b>Minimum and Maximum</b>     | not applicable   |
| <b>Dependencies</b>            | See the Dependencies section of "Send Traps through PATROL SNMP Master Agent" on page 15-13. |
| <b>Recommendation</b>          | none   |

## Ignoring Return IPAddresses

By default, when the PATROL Agent sends an SNMP get request, the PATROL SNMP Master Agent tracks the IP address that the request was sent to. If the response does not return from the same IP Address, the agent ignores the message.

You can use the `/AgentSetup/snmp/ignoreIPAddress` configuration variable to override this default behavior of the PATROL Agent. If this variable is set to **YES** the agent will ignore the IP Address when it gets the response. If the response returns from an IP Address other than where the request was sent, the agent accepts the response.

|                                |   |
|--------------------------------|---|
| <b>Format and Type of Data</b> | text string (yes, no)   |
| <b>Default Value</b>           | none (not defined)  |
| <b>Minimum and Maximum</b>     | none  |
| <b>Dependencies</b>            | none  |
| <b>Recommendation</b>          | Setting this variable can affect the performance of the PATROL Agent. |

---

**Note**

---

This variable is not available in the **config.default** file. You must create this variable manually by using the `wconfig` (Windows) or `xpconfig` (Unix) utility.

---

Once this variable is set, you must reinitialize the agent to apply the variable.

## Configuring the PATROL Event Manager to Send SNMP Traps

The PATROL Event Manager (PEM) associates the individual SNMP trap configuration settings with each event class. This applies to both the Standard Event Catalog and any application-specific event catalog created for a Knowledge Module application.

For each event class, the settings of `NO_TRAP` or `SEND_TRAP` has been added to specify whether the agent will send an SNMP trap when the event is created. This allows more control over the traps. However, you have little control over the format of the SNMP traps. For example, you can not control the event-specific sub ID or the enterprise ID used.

## List of Standard Event Classes

Table 15-1 lists all the standard event classes. These event classes can be useful for sending SNMP traps in other situations, such as a console disconnecting.

**Table 15-1 Standard Event Classes for Sending SNMP Traps (Part 1 of 4)**

| <b>Event Class</b> | <b>Meaning</b>  | <b>Default Trap Setting</b> | <b>Often Used for Traps</b> |
|--------------------|---|-----------------------------|-----------------------------|
| Diag               | Diagnosis event.  | NO_TRAP                     |                             |
| Disconnect         | Console disconnected from agent.  |                             |                             |
| EventArchive       | Events have been archived.  |                             |                             |
| RemPsl             | Used by remote PSL execution.   |                             |                             |
| Result             | Used by remote PSL execution.   |                             |                             |
| PslSet             | Used for remote PSL set execution.  |                             |                             |
| R3FilterData       | Used by the SAP R/3 KM only.  |                             |                             |
| RegApp             | New KM class is now registered and running in the agent (e.g. When a new console connects requesting the KMs that it is interested in viewing). | NO_TRAP                     |                             |
| RemProcess         | Used in remote PSL file transfer and the API.   |                             |                             |
| Unload             | KM class was unloaded by agent.   | SEND_TRAP                   |                             |

**Table 15-1 Standard Event Classes for Sending SNMP Traps (Part 2 of 4)**

| <b>Event Class</b> | <b>Meaning</b>  | <b>Default Trap Setting</b> | <b>Often Used for Traps</b> |
|--------------------|---|-----------------------------|-----------------------------|
| UnregAllApp        | Unregister all applications.  | NO_TRAP                     |                             |
| UpdAppState        | New or updated application state.   |                             | yes                         |
| UpdInstState       | New or updated instance state.  |                             | yes                         |
| UpdParState        | New or updated parameter state.   |                             | yes                         |
| UpdMachineState    | New or updated state for the entire agent (due to some change in the state of an application).  |                             | yes                         |
| WorstApp           | This application now has the worst state of all applications in the agent.  |                             |                             |
| 1                  | Agent's overall state has changed for this agent machine.   |                             |                             |
| 2                  | Worst application class name is provided in this event when the agent's state has changed.  |                             |                             |
| 3                  | Worst application instance name is provided in this event when the agent's state has changed.   |                             |                             |
| 4                  | Discovery has been started for a KM class.  |                             |                             |
| 5                  | Discovery has been disabled for a KM class.   |                             |                             |
| 6                  | Agent and console have different version of a KM.   |                             |                             |
| 7                  | Successful connection to the agent by a user. (i.e. A normal console connection or one involving the API or PSL remote functions).  |                             |                             |
| 9                  | Alarm is cancelled because the condition regarding the parameters violating its thresholds has disappeared. In other words, the parameter's value is no longer a bad value that causes an alarm, and the parameter is going back to the OK state. | SEND_TRAP                   | yes                         |
| 10                 | Recovery action has been executed for the parameter.  | SEND_TRAP                   | yes                         |

**Table 15-1 Standard Event Classes for Sending SNMP Traps (Part 3 of 4)**

| <b>Event Class</b> | <b>Meaning</b>   | <b>Default Trap Setting</b> | <b>Often Used for Traps</b> |
|--------------------|--|-----------------------------|-----------------------------|
| 11                 | Parameter value has exceeded the alarm range thresholds. This will raise a warning or alarm state for this parameter.  | SEND_TRAP                   | yes                         |
| 12                 | All recovery actions have executed and failed to resolve the problem. The parameter will stay in its current state. Agent will not execute any more recovery actions for this parameter. | NO_TRAP                     |                             |
| 13                 | Suspended all parameters of this KM class.   |                             |                             |
| 14 or 15           | Restarting all local and global parameters of the KM class.  |                             |                             |
| 16                 | Parameter description has been modified (i.e. KM editing) and the parameter state is reset to OK.  |                             |                             |
| 17 or 18           | Global parameter has started.  |                             |                             |
| 19                 | Local parameter has started.   |                             |                             |
| 20                 | Parameter had bad output. For example, PSL set on "value" did not provide an integer to a graph or gauge parameter.  |                             |                             |
| 21                 | Local parameter is suspended and will no longer run.   |                             |                             |
| 22 or 23           | Global parameter is suspended and will no longer run.  |                             |                             |
| 24                 | Agent process cache cycle changed.   |                             |                             |
| 25                 | Agent process cache cycle changed.   |                             |                             |

**Table 15-1 Standard Event Classes for Sending SNMP Traps (Part 4 of 4)**

| <b>Event Class</b> | <b>Meaning</b>   | <b>Default Trap Setting</b> | <b>Often Used for Traps</b> |
|--------------------|--|-----------------------------|-----------------------------|
| 26 or 27           | Application discovery is disabled for this KM class.   | NO_TRAP                     |                             |
| 28                 | Username/password were invalid to connect to the agent (e.g. through the API or PSL remote functions). |                             |                             |
| 29                 | Internal agent or PEM failure of some type.  |                             |                             |
| 38                 | Parameters of a KM were restarted.   |                             |                             |
| 39                 | Parameter threshold was exceeded by parameter value. State change event.                               |                             |                             |
| 40                 | PSL response-related event. Created when a PSL response function is launched by the agent.             |                             |                             |
| 41, 42, or 43      | Information event. Placeholder for user-defined events. Not generated internally by the agent.         |                             |                             |

# Sending SNMP Traps Based on PATROL Events

To use the agent to send SNMP traps based on PATROL events, you must enable them in the agent configuration.

## Destination of SNMP PEM-based Traps

The recipient list of SNMP traps is set in the agent configuration. The `/snmp/piV1m_list` variable contains a comma separated list of host names and/or IP addresses, which represent SNMP trap destinations. The PATROL SNMP Subagent uses this variable with the `/snmp/trapMibTable` variable to bypass the PATROL SNMP Master Agent and send traps directly to SNMP Managers stored in this variable.

---

### Note

---

This list of trap destinations does not affect the recipients of SNMP traps sent by PSL `snmp_trap_send()`.

---

The `/snmp/piV1m_list` configuration variable specifies which SNMP managers receive traps from the PATROL Agent.

|                                |   |
|--------------------------------|---|
| <b>Format and Type of Data</b> | An entry for a manager consists of host name, port number, and read community name. Items in an entry are separated by a forward slash (/). Entries must be separated with a comma.<br><br>host/port/r_community_string |
| <b>Default Value</b>           | "" (empty string = no managers)   |
| <b>Minimum and Maximum</b>     | not applicable  |
| <b>Dependencies</b>            | /snmp/trapMibTable uses this list to send traps directly to SNMP Managers listed in this variable   |
| <b>Recommendation</b>          | The port number should correspond to the /snmp/trap_port.   |

---

**Example**

---

```
"/snmp/piV1m_list" = {
REPLACE="nasa/162/public,198.207.223.112/162/public" }
```

---

## SNMP Support Based on PATROL Event Manager

The `/AgentSetup/pemSnmpSupport` configuration variable determines whether PATROL Events trigger SNMP events.

|                                |                    |
|--------------------------------|--------------------|
| <b>Format and Type of Data</b> | boolean, yes or no |
| <b>Default Value</b>           | yes                |
| <b>Minimum and Maximum</b>     | not applicable     |
| <b>Dependencies</b>            | none               |
| <b>Recommendation</b>          | none               |

## Filter Traps Based on PATROL Event Severity Level

The `/AgentSetup/pemPFSnmpNSeverity` configuration variable specifies the severity level that triggers the PATROL Agent to send an SNMP trap. The agent will send traps for only those events with a numerical severity higher than or equal to the variable value.

|                            |  |
|----------------------------|--|
| <b>Values</b>              | <p>The numerical severity of the event. Only events with numerical severity higher or equal to the specified numbers will trigger SNMP traps.</p> <ul style="list-style-type: none"><li><b>0</b>—(no label)</li><li><b>1</b>—INFORMATION</li><li><b>2</b>—STATE_CHANGE</li><li><b>3</b>—WARNING</li><li><b>4</b>—ALARM</li><li><b>5</b>—RESPONSE</li></ul> <p>The most severe is level is five (5). Zero (0) sends an event to the PATROL Agent, but the event is not displayed in the PATROL Event Manager.</p> |
| <b>Default Value</b>       | 1  |
| <b>Minimum and Maximum</b> | not applicable   |
| <b>Dependencies</b>        | none   |
| <b>Recommendation</b>      | none   |

## Specify Application That Issued the Trap

The `/AgentSetup/pemPFSnmpOrigin` configuration variable specifies the application where the event occurred.

|                                |                                       |
|--------------------------------|---------------------------------------|
| <b>Format and Type of Data</b> | text string, application name         |
| <b>Default Value</b>           | "" (empty string for any application) |
| <b>Minimum and Maximum</b>     | not applicable                        |
| <b>Dependencies</b>            | none                                  |
| <b>Recommendation</b>          | none                                  |

## Specify the Node Name Where the Trap Originated

The `/AgentSetup/pemPFSnmpNode` configuration variable specifies the name of the node where the event occurred. Presently, this variable does not have any effect because all events originate from the agent's node. It is reserved for future use.

|                                |                        |
|--------------------------------|------------------------|
| <b>Format and Type of Data</b> | text string, node name |
| <b>Default Value</b>           | "" (any node)          |
| <b>Minimum and Maximum</b>     | not applicable         |
| <b>Dependencies</b>            | none                   |
| <b>Recommendation</b>          | none                   |

## Filter Traps Based on PATROL Event ID

The `/AgentSetup/pemPFSnmpEidRange` configuration variable specifies the range of event IDs to filter out.

|                            |  |
|----------------------------|--|
| <b>Values</b>              | <b>x</b> a single ID x<br><b>x/y</b> any value between and including x, y<br><b>-/y</b> any positive value equal or less than y<br><b>x/-</b> any positive value equal or greater than x<br>x and y are positive cardinal value smaller than xFFFFFFFF.<br><b>-/-</b> any positive cardinal value smaller than xFFFFFFFF |
| <b>Default Value</b>       | "" (do not filter out by event id)   |
| <b>Minimum and Maximum</b> | not applicable   |
| <b>Dependencies</b>        | none   |
| <b>Recommendation</b>      | none   |

## Filter Traps Based on PATROL Event Class

The `/AgentSetup/pemPFSnmpEvClass` configuration variables specifies the event classes whose traps agent will filter out and not send.

|                                |                                       |
|--------------------------------|---------------------------------------|
| <b>Format and Type of Data</b> | text string, event class name         |
| <b>Default Value</b>           | "" (do not filter out by event class) |
| <b>Minimum and Maximum</b>     | not applicable                        |
| <b>Dependencies</b>            | none                                  |
| <b>Recommendation</b>          | none                                  |

## Filter Events Based on PATROL Event Description

The `/AgentSetup/pemPFSnmpPattern` configuration variable specifies the pattern that is compared against the description of an event. If the pattern matches a portion of the description, the agent will filter out the event's trap and not send it.

|                                |                                       |
|--------------------------------|---------------------------------------|
| <b>Format and Type of Data</b> | text string, pattern                  |
| <b>Default Value</b>           | "" (do not filter out by description) |
| <b>Minimum and Maximum</b>     | not applicable                        |
| <b>Dependencies</b>            | none                                  |
| <b>Recommendation</b>          | none                                  |

## Specify a Time Period to Send Traps Based on PATROL Events

To send traps based on PATROL Events during a particular time period, you can set `/AgentSetup/pemPFSnmpStartTime` and `/AgentSetup/pemPFSnmpEndTime` configuration variables.

## Start Time

The `/AgentSetup/pemPFSnmpStartTime` configuration variable specifies the start of the monitoring period.

|                                |   |
|--------------------------------|---|
| <b>Format and Type of Data</b> | <b>MMddhhmm[yy]</b><br><b>MM</b> —month<br><b>dd</b> —day<br><b>hh</b> —hour<br><b>mm</b> —minutes<br><b>yy</b> —years (optional) |
| <b>Default Value</b>           | "" (all the time)   |
| <b>Minimum and Maximum</b>     | not applicable  |
| <b>Dependencies</b>            | AgentSetup/pemPFSnmpEndTime specifies the end of the monitoring period  |
| <b>Recommendation</b>          | none  |

## End Time

The `/AgentSetup/pemPFSnmpEndTime` configuration variable specifies the end of the monitoring period.

|                                |   |
|--------------------------------|---|
| <b>Format and Type of Data</b> | <b>MMddhhmm[yy]</b><br><b>MM</b> —month<br><b>dd</b> —day<br><b>hh</b> —hour<br><b>mm</b> —minutes<br><b>yy</b> —years (optional) |
| <b>Default Value</b>           | "" (all the time)   |
| <b>Minimum and Maximum</b>     | not applicable  |
| <b>Dependencies</b>            | AgentSetup/pemPFSnmpStartTime specifies the start of the monitoring period  |
| <b>Recommendation</b>          | none  |

## Filter Traps Based on PATROL Event Type

The `/AgentSetup/pemPFSnmpTypeMask` configuration variable determines which PATROL events send traps based on the event's type. The filter is inclusive. It sends traps for all types listed in the type mask variable.

|                                |   |
|--------------------------------|---|
| <b>Format and Type of Data</b> | Text string containing one or more types separated by commas<br><br>Types of PATROL events include<br>I—INFORMATION<br>S—CHANGE_STATUS<br>E—ERROR<br>W—WARNING<br>A—ALARM<br>R—RESPONSE |
| <b>Default Value</b>           | I, S, E, W, A, R (send for all types)   |
| <b>Minimum and Maximum</b>     | not applicable  |
| <b>Dependencies</b>            | none  |
| <b>Recommendation</b>          | none  |

## Filter Traps Based on PATROL Event Status

The `/AgentSetup/pemPFSnmpStatusMask` configuration variable determines which PATROL events send traps based on the event's status. The filter is inclusive. It sends traps for all statuses listed in the status mask variable.

|                                |   |
|--------------------------------|---|
| <b>Format and Type of Data</b> | Text string containing one or more statuses separated by commas<br><br>Statuses include<br><b>O</b> —OPEN<br><b>A</b> —ACKNOWLEDGED<br><b>C</b> —CLOSED<br><b>E</b> —ESCALATED<br><b>D</b> —DELETED |
| <b>Default Value</b>           | O, A, C, E, D (send for all statuses)   |
| <b>Minimum and Maximum</b>     | not applicable  |
| <b>Dependencies</b>            | none  |
| <b>Recommendation</b>          | none  |

# Set SNMP Trap Format

The PATROL Agent can send SNMP trap information in two different formats. To select a format, set the variable of the desired format to **yes** and set the variable of the other format to **no**.

---

**Note**

---

If both variables are set to **yes** for a single event, the agent issues two SNMP traps: one in V3.0 format and one in V3.1 format.

---

## 3.1 Format

The `/AgentSetup/pemIssueV31traps` configuration variable determines whether PATROL uses version 3.1 formats to send SNMP traps.

|                                |                    |
|--------------------------------|--------------------|
| <b>Format and Type of Data</b> | boolean, yes or no |
| <b>Default Value</b>           | yes                |
| <b>Minimum and Maximum</b>     | not applicable     |
| <b>Dependencies</b>            | none               |
| <b>Recommendation</b>          | none               |

## 3.1 Example

This SNMP trap is for PEM Standard Event 11 in 3.1 format.

```
From: 127.11.502.22 (1.6.3.4.1.4.1201.1.1.7) Enterprise Specific (5)
Uptime: 0 day(s) 0:13:16 (79686)
1.6.3.4.1.4.1201.1.1.7.1.0 85 string Alarm #2 of global
parameter 'parm' triggered on 'PSLCOM.PSLCOM'. 3 <= 3.00 <= 3
1.6.3.4.1.4.1201.1.1.7.2.0 23 string /PSLCOM/PSLCOM/parm
1.6.3.4.1.4.1201.1.1.7.3.0 0 string
```

### 3.0 Format

The `/AgentSetup/pemIssueV30traps` configuration variable determines whether PATROL uses version 3.0 formats to send SNMP traps.

|                                |                    |
|--------------------------------|--------------------|
| <b>Format and Type of Data</b> | boolean, yes or no |
| <b>Default Value</b>           | no                 |
| <b>Minimum and Maximum</b>     | not applicable     |
| <b>Dependencies</b>            | none               |
| <b>Recommendation</b>          | none               |

### 3.0 Example

This SNMP trap is for PEM Standard Event 11 in 3.0 format.

```
From: 127.11.502.22 (1.6.3.4.1.4.1201.1.1.7) Enterprise Specific (11)
Uptime: 0 day(s) 0:18:08 (108876)
1.6.3.4.1.4.1201.1.1.7.1.0 29 string /PSLCOM/PSLCOM/parm ALARM
```

# Disabling SNMP Trap Support for PATROL Events

PATROL provides the following methods for preventing the agent from sending SNMP trap information based on PATROL Events:

- Delete all recipients listed in the SNMP Managers' list variable, psV1m\_list.
- Set the /AgentSetup/pemSnmpSupport variable to **no**.
- Set the /snmp/support variable to **no**.

---

### **Note**

---

In the PATROL Agent, you can disable SNMP trap sending based on PATROL Events without affecting other SNMP functionality. For example, you can still use the PSL SNMP get, set, get-next, or walk on other host MIBs even if traps are disabled.

---



---

# SNMP Configuration and Implementation Using PSL

This chapter provides an overview of PSL support for SNMP. This chapter contains the following sections:

|  |       |
|--|-------|
| Configuring the PATROL Agent as an SNMP Manager . . . . .              | 16-2  |
| Listening for SNMP Traps . . . . .                                     | 16-2  |
| Getting and Setting MIB Variables . . . . .                            | 16-3  |
| Sending SNMP Traps Based on PSL Functions. . . . .                     | 16-4  |
| Overview of PSL Functions for SNMP . . . . .                           | 16-4  |
| Set Default Community Name for PSL SNMP Functions. . . . .             | 16-5  |
| Managing PSL SNMP Built-In Functions Execution . . . . .               | 16-6  |
| PSL SNMP Built-In Functions . . . . .                                  | 16-8  |
| Sending SNMP Traps. . . . .  | 16-9  |
| Debugging PSL Functions for SNMP . . . . .                             | 16-9  |
| Interpreting Error Messages from PSL Functions . . . . .               | 16-10 |
| Distinguishing Instance State Changes from Propagated States . . . . . | 16-11 |
| Gathering PATROL Data from PATROL MIB Tables . . . . .                 | 16-12 |

# Configuring the PATROL Agent as an SNMP Manager

Through PSL functions, the PATROL Agent can act as an SNMP Manager by listening for traps and getting and setting information stored in SNMP agent variables, referred to as MIB variables.

Table 16-1 lists PSL functions that support this method of SNMP trap receiving and analysis.

## Listening for SNMP Traps

During trap listening, the PATROL Agent works as an SNMP manager that receives and handles traps from SNMP agents, including itself.

Table 16-1 lists the PSL function to use for the task you want to perform.

**Table 16-1 Functions for SNMP Manager/Trap Handler**

| <b>Task to be Performed</b>  | <b>PSL Function to Use</b>       |
|--|----------------------------------|
| start accumulating incoming traps                                    | <code>snmp_trap_listen()</code>  |
| capture the arriving traps   | <code>snmp_trap_receive()</code> |
| close a trap socket and ignore all unprocessed and/or arriving traps | <code>snmp_trap_ignore()</code>  |

## Getting and Setting MIB Variables

The PATROL Agent can act as an SNMP Manager by getting and setting variables inside SNMP agents through PSL functions. Table 16-2 lists the function to use for the task you want to perform.

**Table 16-2 Functions for Getting and Setting MIB Variables**

| <b>Task You Want to Perform</b>  | <b>PSL Function to Use</b>  |
|--|---|
| open a session to an SNMP agent by locating the host and creating an internal structure with default information                                     | <code>snmp_open()</code>  |
| close the session with SNMP agent  | <code>snmp_close()</code>   |
| list SNMP sessions that are currently open, return default parameters for a specific snmp session, or alter the default settings for an SNMP session | <code>snmp_config()</code>  |
| fetch MIB variables from an SNMP agent   | <code>snmp_get()</code> ,<br><code>snmp_get_next()</code> , or<br><code>snmp_walk()</code><br><br>You can also use <code>snmp_h_*</code> functions. The <code>snmp_h_*</code> functions accept host name instead of session and automatically open and close the session. |
| set MIB variables  | <code>snmp_set()</code><br><br>You can also use <code>snmp_h_*</code> functions. The <code>snmp_h_*</code> functions accept host name instead of session and automatically open and close the session.  |

---

**Note**

---

`snmp_h_*` functions use the default port, but can be configured to use a different port.

---

# Sending SNMP Traps Based on PSL Functions

Using PSL for sending SNMP traps allows you to control

- under what circumstances SNMP traps are sent
- the format the SNMP trap uses
- destination of the SNMP trap

---

**Note**

---

BMC Software recommends using a PSL Notification command in the event catalog rather than a recovery action, because it can be done generically for all parameters rather than adding a recovery action to each one.

---

This section tells you how you can use PSL to control how the PATROL SNMP Master Agent and the PATROL Agent interact with SNMP.

## Overview of PSL Functions for SNMP

The primary groups of PSL functions for SNMP are

- listening for traps
- sending traps
- starting and stopping the SNMP sub-agent
- getting and setting Management Information Base (MIB) variables
- changing the registered SNMP manager list
- debugging

PSL functions allow you to manage a number of processes, including starting and stopping the PATROL SNMP Sub-Agent and changing the list of registered SNMP managers.

Some of these PSL functions are briefly described in this section. Refer to the *PATROL Script Language Reference Manual* for detailed information about all PSL functions for SNMP.

## Set Default Community Name for PSL SNMP Functions

The agent allows you to assign the read and write default community name used if one is not provided when starting a SNMP session with `snmp_open()`.

### Read Functions

The `/snmp/default_r_community` agent configuration variable sets the default read community name for the built-in PSL SNMP function, `snmp_open()`, if a community name is not provided as an argument.

|                                |                             |
|--------------------------------|-----------------------------|
| <b>Format and Type of Data</b> | text string, not applicable |
| <b>Default Value</b>           | public                      |
| <b>Minimum and Maximum</b>     | not applicable              |
| <b>Dependencies</b>            | none                        |
| <b>Recommendation</b>          | none                        |

## Write Functions

The `/snmp/default_w_community` agent configuration variable sets the default community name for built-in PSL SNMP functions such as `snmp_set( )`.

|                                |                             |
|--------------------------------|-----------------------------|
| <b>Format and Type of Data</b> | text string, not applicable |
| <b>Default Value</b>           | private                     |
| <b>Minimum and Maximum</b>     | not applicable              |
| <b>Dependencies</b>            | none                        |
| <b>Recommendation</b>          | none                        |

## Managing PSL SNMP Built-In Functions Execution

The agent allows you to manage certain aspects of how the PSL SNMP Built-In Functions execute. You can determine the number of times the agent attempts to execute a function, how long the agent waits for a function to return a value, on which port number the function is executed.

### Timeout

The `/snmp/default_timeout` agent configuration variable determines how long the agent waits for a response to a built-in PSL SNMP function before it reports that the function did not return a value.

|                                |                       |
|--------------------------------|-----------------------|
| <b>Format and Type of Data</b> | numeric, milliseconds |
| <b>Default Value</b>           | 500                   |
| <b>Minimum and Maximum</b>     | not applicable        |
| <b>Dependencies</b>            | none                  |
| <b>Recommendation</b>          | none                  |

## Retries

The `/snmp/default_retries` agent configuration variable limits the number of times that the agent attempts to execute a built-in PSL SNMP function before it reports that the function failed.

|                                |                   |
|--------------------------------|-------------------|
| <b>Format and Type of Data</b> | numeric, attempts |
| <b>Default Value</b>           | 3                 |
| <b>Minimum and Maximum</b>     | not applicable    |
| <b>Dependencies</b>            | none              |
| <b>Recommendation</b>          | none              |

## Port

The `/snmp/default_port` agent configuration variable sets the port number used by the built-in PSL SNMP function, `snmp_open( )`, if a port number is not provided as an argument.

|                                |                      |
|--------------------------------|----------------------|
| <b>Format and Type of Data</b> | numeric, port number |
| <b>Default Value</b>           | 161                  |
| <b>Minimum and Maximum</b>     | not applicable       |
| <b>Dependencies</b>            | none                 |
| <b>Recommendation</b>          | none                 |

## PSL SNMP Built-In Functions

PSL contains a variety of functions that support SNMP. For more detailed information about these functions, see the *PATROL Script Language Reference Manual*.

Table 16-3 lists these functions and provides a brief description of each. The list includes SNMP Agent and SNMP Manager functions.

**Table 16-3 PSL Functions for SNMP Support (Part 1 of 2)**

| <b>PSL Built-In Function</b> | <b>Purpose</b>   |
|------------------------------|--|
| snmp_agent_config()          | Verify if SNMP support is active or inactive.                                |
| snmp_agent_start()           | Start the agent's SNMP sub-agent.  |
| snmp_agent_stop()            | Stop the agent's SNMP sub-agent.   |
| snmp_close()                 | Close a session used for SNMP get/set access.                                |
| snmp_config()                | Report on sessions currently open.   |
| _snmp_debug()                | Control internal agent SNMP debugging features.                              |
| snmp_get()                   | Perform an SNMP get action on a MIB object, using an open session.           |
| snmp_get_next()              | Perform an SNMP "get next" action on a MIB object, using an open session.    |
| snm_h_set()                  | Perform an SNMP set action on a MIB object using a brief session.            |
| snmp_h_get_next()            | Perform an SNMP "get next" action on a MIB object using a brief session.     |
| snmp_h_set()                 | Perform an SNMP set action on a MIB object using a brief session.            |
| snmp_open()                  | Open a new session to be used for get, set, and get next operations.         |
| snmp_set()                   | Perform an SNMP set action on a MIB object, using an open session.           |
| snmp_trap_ignore()           | Ignore traps incoming to the PATROL SNMP sub-agent.                          |
| snmp_trap_listen()           | Start recording incoming traps to the PATROL SNMP sub-agent.                 |
| snmp_raise_std_trap()        | Raise the standard PATROL trap.  |
| snmp_trap_receive()          | Read the traps that were accumulated from incoming traps to the agent.       |
| snmp_trap_register_im()      | Change or read the list of SNMP managers registered to receive PATROL traps. |

**Table 16-3 PSL Functions for SNMP Support (Part 2 of 2)**

| PSL Built-In Function         | Purpose  |
|-------------------------------|--|
| <code>snmp_trap_send()</code> | Send an SNMP trap with full control over its format.           |
| <code>snmp_walk()</code>      | Perform an SNMP walk operation on a MIB using an open session. |

## Sending SNMP Traps

During trap sending, the PATROL Agent works in an SNMP agent role. Table 16-4 lists the function to use for the task you want to perform.

**Table 16-4 Functions for Sending Traps**

| Task You Want to Perform  | PSL Function to Use                           |
|---|---|
| send any traps to any given SNMP manager  | <code>snmp_trap_send()</code>                 |
| send the trap <code>patrolTrapVlRaised</code> , with <code>patrolTrapText.0</code> in a packet, to all entities registered in the <code>prVlmTable</code> | <code>snmp_trap_raise_std_trap("text")</code> |

## Debugging PSL Functions for SNMP

Use the `snmp_debug(flags)` function to debug the PSL you write. The `snmp_debug(flags)` function accepts a binary flag (0, 1, 2, or 3) that activates PSL SNMP debugging features. It returns the old settings or NULL indicating an error. Table 16-5 lists the function to use for the task you want to perform.

**Table 16-5 Functions for Debugging PSL Functions**

| Task You Want to Perform   | <code>snmp_debug(flags)</code> Function to Use |
|--|--|
| dump all in/out packets on <code>stdout</code> when the agent is in no-daemon mode     | <code>snmp_dump_packet(1)</code>               |
| get error information that may not be reported to the console window, such as timeouts | <code>snmp_report_error(2)</code>              |

# Interpreting Error Messages from PSL Functions

Table 16-6 describes global error messages for PSL functions for SNMP. They are considered *global* because any SNMP PSL function can generate one of these messages.

**Table 16-6 Global Error Messages for SNMP PSL Functions**

| Error Message                | Description   |
|------------------------------|---|
| E_PSL_BAD_FUNCTION_PARAMETER | A function fails to parse a parameter, which could be caused, for example, by a bad address or trap definition. |
| E_PSL_SNMP_ERROR             | A function tries to send or receive an invalid packet to or from another SNMP entity.                           |
| E_PSL_SNMP_NOT_SUPPORTED     | SNMP support is turned off.   |
| NULL                         | If an error occurs, a function returns a null string or “ ”.  |

When an error occurs, the user does not see any of the error messages in Table 16-6. A user sees nothing since all SNMP PSL functions return the NULL string after encountering an error. A user can determine which error occurred most recently by displaying or printing the value of the PATROL PSL error variable. This variable holds an integer that corresponds to one of the error messages above.

The *PATROL Script Language Reference Manual* provides more information on working with error messages.

# Distinguishing Instance State Changes from Propagated States

PATROL has two ways an application instance can achieve the WARN or ALARM state

- a PSL change\_state
- an inherited parameter state

The WARN and ALARM state can be detected by analyzing the “status” and “ruleState” variables in the agent symbol table. This is possible through a PSL script, but is also possible through external SNMP access to the PATROL MIB.

The OFFLINE state can only occur due to a PSL change\_state, and cannot be inherited from parameter states.

The “status” and “ruleState” variables are built-in variables in the instance's symbol table. Table 16-7 lists those situations that apply.

**Table 16-7 “status” and “ruleState” Variables**

| <b>“status”</b> | <b>“ruleState”</b> | <b>Meaning</b>  |
|-----------------|--------------------|---|
| OK              | OK                 | no activity such as alarms or state changes             |
| WARN            | OK                 | propagated WARN parameters                              |
| ALARM           | OK                 | propagated ALARM parameters                             |
| OK              | WARN               | not possible  |
| WARN            | WARN               | PSL change_state to WARN                                |
| ALARM           | WARN               | PSL change_state to WARN, but also a parameter in ALARM |
| OK              | ALARM              | not possible  |
| WARN            | ALARM              | not possible  |
| ALARM           | ALARM              | PSL change_state to ALARM                               |
| any state       | OFFLINE            | PSL change_state to OFFLINE                             |

# Gathering PATROL Data from PATROL MIB Tables

The PATROL Agent MIB consists of several dynamic tables that allow you to navigate PATROL KM parameter data collected using SNMP get operations. However, the dynamic nature of object IDs make it difficult to poll for some applications. Using a separate values table with the parameter name allows more static access to the same parameter data.

Using an SNMP get operation, it is possible to obtain many attributes of a parameter, instance, or application class object. Some of these attributes include

- parameter value—SNMP get on the “value” variable of a parameter name
- parameter status—SNMP get on the “status” variable of a parameter name
- parameter last set time—SNMP get on the “time” variable of a parameter name
- instance status—SNMP get on the “status” variable at instance level
- application class status—SNMP get on the “status” variable at class level

See AppendixB, “Changing Variables in a File Manually,” for more information on tables that map PATROL managed objects to SNMP MIB variables.

---

### Note

---

Using an SNMP get operation you can only receive the current value and statistics of a parameter. It is not possible to receive the history of a parameter using SNMP get.

---

---

# Managing Parameter Overrides

This chapter provides an overview of the different methods of overriding parameters and explains how to use an external file to manage parameter overrides. This chapter discusses the following topics:

|  |       |
|--|-------|
| Parameter Properties that Can Be Overriden . . . . .                             | 17-2  |
| Methods of Overriding Parameters. . . . .  | 17-2  |
| External File Overrides . . . . .  | 17-3  |
| Operator Overrides. . . . .  | 17-3  |
| PSL Overrides . . . . .  | 17-3  |
| Developer Overrides . . . . .  | 17-3  |
| Controlling Which Override Methods Are Used . . . . .                            | 17-4  |
| If External File Overrides Are Disabled . . . . .                                | 17-5  |
| If Operator Overrides Are Disabled. . . . .                                      | 17-5  |
| If PSL Overrides Are Disabled . . . . .  | 17-5  |
| Combining Multiple Methods of Overriding Parameters. . . . .                     | 17-6  |
| Using Developer Overrides . . . . .  | 17-6  |
| Using External File Overrides and Operator Overrides. . . . .                    | 17-6  |
| Using External File Overrides and PSL Overrides . . . . .                        | 17-7  |
| Using Operator Overrides and PSL Overrides. . . . .                              | 17-7  |
| Using External File Overrides, Operator Overrides, and<br>PSL Overrides. . . . . | 17-7  |
| Using External Override Files to Manage Parameter Overrides . . .                | 17-8  |
| Storing Overrides in a Single File . . . . .                                     | 17-8  |
| Storing Overrides in Multiple Files . . . . .                                    | 17-9  |
| External Override File Location and Privileges. . . . .                          | 17-11 |
| External Override File Names . . . . .   | 17-11 |
| Defining Which Override Files to Use . . . . .                                   | 17-12 |
| Using Macro Variables in Override Files. . . . .                                 | 17-13 |

|   |       |
|---|-------|
| External Override File Poll Cycle .....   | 17-13 |
| External Override File Format .....       | 17-14 |
| Deleting the External Override File ..... | 17-17 |
| Additional Functions .....                | 17-17 |

## Parameter Properties that Can Be Overriden

The baseline properties of a parameter are part of the parameter's definition in its Knowledge Module file. Although these baseline properties might be appropriate for most systems in your environment, you might have some systems which are exceptions. You can override the following properties of a parameter:

- whether or not the parameter is active
- the polling interval for the parameter
- border, alarm1, and alarm2 ranges

A parameter override cannot change any other part of a parameter, such as its type, command, output, or help topic.

## Methods of Overriding Parameters

You can override parameters using any of the following methods:

- external file overrides
- operator overrides
- PSL overrides
- developer overrides

## External File Overrides

This is the recommended method for managing parameter overrides. Overrides are stored in external files with the PATROL Agent. These files are easy to distribute and replace. For more information, see “Using External Override Files to Manage Parameter Overrides” on page 17-8.

## Operator Overrides

An operator override is stored in an external database. This method works best for small sites. For more information see the *PATROL Console for Unix User Guide* or *PATROL Console for Microsoft Windows 2000 User Guide, III—Customizing PATROL*.

## PSL Overrides

The PSL function `set_alarm_ranges()` changes the alarm ranges of a parameter. This method is useful for intelligent Knowledge Modules that determine their own alarm ranges during run-time. However, this method is CPU intensive and can cause problems if parameters are changed, so it is not recommended for regular use. For more information see the *PATROL Script Language Reference Manual Volume 2, PSL Functions*.

## Developer Overrides

A developer override is a change to a parameter for a specific application on a specific host. It is stored in a special section of the KM file and can be created by only a developer console. This method is not recommended for large enterprises. For more information see the *PATROL Console for Unix User Guide* or *PATROL Console for Microsoft Windows 2000 User Guide, III—Customizing PATROL*.

# Controlling Which Override Methods Are Used

You can use the **patrol.conf** file to selectively enable and disable external file overrides, operator overrides, and PSL overrides.

The location of the **patrol.conf** file depends on the operating system. On Unix, it is in the **\$PATROL\_HOME/etc/patrol.d** directory. On Windows, it is in the PATROL Security directory specified in PATROL Agent Configuration Utility.

For more information about the **patrol.conf** file, see the *PATROL Console for Unix User Guide* or *PATROL Console for Microsoft Windows 2000 User Guide, III—Customizing PATROL*.

The following variables in the **agentrights** section of the **patrol.conf** file control which methods of implementing overrides are allowed and which are disabled.

- **allowexternaloverride** can have a value of true or false, which allows or disables external file overrides.
- **allowpsloverride** can have a value of true or false, which allows or disables the PSL overrides.
- **allowoperatoroverride** can have a value of true or false, which allows or disables operator overrides. This setting takes precedence over any setting on the console.

The following is an example of the **agentrights** section of the **patrol.conf** file that allows all three types of overrides:

---

```
[AGENT]
define agentrights allrights
allowexternaloverride=true
allowpsloverride=true
allowoperatoroverride=true
```

```
end
agent *, allrights
```

---

---

**Note**

You must manually define *allowexternaloverride* parameter in the **patrol.conf** file, it is not included in the file by default.

---

---

**Note**

You cannot disable developer overrides.

---

## If External File Overrides Are Disabled

Even if an external file is specified, it is not used.

## If Operator Overrides Are Disabled

If someone running an operator console attempts to override a parameter, an error message is displayed indicating that the agent does not allow the override.

## If PSL Overrides Are Disabled

If the PSL function `set_alarm_ranges()` is run, it fails and PATROL displays the following error message:  
“set disallowed by agent rights”.

# Combining Multiple Methods of Overriding Parameters

This section describes how the PATROL Agent deals with conflicts between different methods of overriding parameters.

---

**Tip**

---

Using multiple methods of overriding parameters is not recommended.

---

## Using Developer Overrides

Do not use a developer console to override parameters on an agent that uses any other method of parameter overrides. When defining and testing the baseline properties of a parameter, connect to an agent that has only the Knowledge Module loaded that you want to baseline, and no overrides.

When you are using a developer console, the console reflects the settings in the Knowledge Module until you open a parameter with overrides. When you open a parameter with overrides, the console reflects the override settings, and when you save the Knowledge Module the override settings are saved.

## Using External File Overrides and Operator Overrides

If an operator tries to override a parameter and an external file override already exists for it, the operator will get a message on the console stating that the agent does not allow the override.

The overrides in the external file are always used. Operator overrides are only used if the parameter is not already overridden.

## Using External File Overrides and PSL Overrides

Whichever override is applied most recently is used. Normally the most recent override is the PSL command. However, if an external file change is made after the PSL function call, the override in the external file is used until the next time the `set_alarm_ranges()` command is run for that parameter.

## Using Operator Overrides and PSL Overrides

Whichever override is applied most recently is used. Normally the most recent override is the PSL command. However, if an operator override is made after the PSL function call, the operator override is used until the next time the `set_alarm_ranges()` command is run for that parameter.

## Using External File Overrides, Operator Overrides, and PSL Overrides

If an operator tries to override a parameter and an external override already exists, the operator will get a message on the console stating that the agent does not allow the override.

Otherwise, whichever override is applied most recently is used. Normally the most recent override is the PSL command. However, if an operator override or an external file change is made after the PSL function call, that override is used until the next time an override is applied.

# Using External Override Files to Manage Parameter Overrides

Using external files is the recommended method for managing parameter overrides. In this method, parameter overrides are stored in one or more external files. The PATROL Agent periodically checks if an external override file has changed or if a different external override file should be used. If a file or its contents change, the PATROL Agent implements the new overrides.

You can store your override definitions in a single file or in multiple files in the same directory. The following sections explain the procedures and uses for each method.

## Storing Overrides in a Single File

When you store override definitions in a single file, the PATROL Agent periodically checks the timestamp of the file to see if it has been updated. If the file has been updated, the agent implements the new parameter override definitions contained in the file. The following procedure outlines the steps you must perform to store your override definitions in a single file. More details about each step are documented in the rest of this chapter.

- Step 1** Define the `/AgentSetup/ExternalOverride` variable as follows:  
`/AgentSetup/ExternalOverride = myfile`, where *myfile* is the name of the override file you want to use. This file is not required to have an extension.
- Step 2** Create *myfile* with the override definitions you want to use. The filename must be the same filename you define in `/AgentSetup/ExternalOverride`.

The following example file, **myfile**, defines overrides for the `LDldDiskTimePercent` parameter of the `NT_LOGICAL_DISKS` application class and the `PAWorkRateExecsMin` parameter of the `PATROL_NT` application class.

**myfile**

```
[ /NT_LOGICAL_DISKS//LDldDiskTimePercent ]
ACTIVE=1
BORDER_ACTIVE=0
ALARM1_ACTIVE=0
ALARM2_ACTIVE=0

[ /PATROL_NT/PATROL_NT/PAWorkRateExecsMin ]
ACTIVE=1
BORDER_ACTIVE=0
ALARM1_ACTIVE=0
ALARM2_ACTIVE=0
```

Anytime you create or update this file, the PATROL Agent applies the new overrides to the parameters.

For details about creating the content of override files, see “External Override File Format” on page 17-14

## Storing Overrides in Multiple Files

When you store override definitions in multiple files, the PATROL Agent checks the directory containing the files for updated definitions. If the files contain new definitions, the agent implements them. The following procedure outlines the steps you must perform to store your override definitions in multiple files. More details about each step are documented in the rest of this chapter.

- Step 1** Define the */AgentSetup/ExternalOverride* variable as follows:  
*/AgentSetup/ExternalOverride = mydir*, where *mydir* is the relative path to the %PATROL\_HOME%/lib/admin directory.
- Step 2** Create the files with the override definitions you want to use. When using multiple files, each file can only define overrides for a single application class. The filename of each file must exactly match the name of the application class for which it defines overrides.

The following example shows two separate override files, **NT\_LOGICAL\_DISKS** and **PATROL\_NT**. Each file is named after the application class for which it defines parameter overrides. When the */AgentSetup/ExternalOverride* variable is defined as the relative path to the override file directory, the PATROL Agent reads all files in the directory and applies the overrides as defined in each file.

**NT\_LOGICAL\_DISKS**

```
[ /NT_LOGICAL_DISKS//LDldDiskTimePercent ]  
ACTIVE=1  
BORDER_ACTIVE=0  
ALARM1_ACTIVE=0  
ALARM2_ACTIVE=0
```

**PATROL\_NT**

```
[ /PATROL_NT/PATROL_NT/PAWorkRateExecsMin ]  
ACTIVE=1  
BORDER_ACTIVE=0  
ALARM1_ACTIVE=0  
ALARM2_ACTIVE=0
```

For details about creating the content of override files, see “External Override File Format” on page 17-14

**Step 3** Create the **@timestamp** file.

The PATROL Agent checks for a file called **@timestamp** (without a file extension) in the following path: */AgentSetup/ExternalOverride*.

The agent does not read the file, it only checks its timestamp. You must create the **@timestamp** file manually.

If the **@timestamp** file has a different timestamp as before or does not exist, the PATROL Agent reads all files in the directory which have changed.

## External Override File Location and Privileges

All external override files must be located in the \$PATROL\_HOME/lib/admin directory (Unix) or %PATROL\_HOME%\lib\admin folder (Windows). This directory can be read-only.

The PATROL Agent does not write to external override files and it reads them with the PATROL\_ADMIN user. Therefore, the external override files can be read-only and owned by the PATROL\_ADMIN user.

---

### Note

---

You must manually create external override files.

---

## External Override File Names

The file naming conventions for override files vary depending on whether you use a single file or multiple files. Filenames must not have extensions.

### Single File Method

If you are storing your override definitions in a single file, you can use any filename you want; however, you must make sure it matches the filename you define in the *The /AgentSetup/ExternalOverride* configuration variable.

### Multiple File Method

If you store your override definitions in multiple files, each file can specify overrides only for a single application class. Each override file filename must exactly match the name of the application class that it defines overrides for.

If a file contains a definition for an application class that is different from the filename, the PATROL Agent ignores the definition. For example, a file named **CPU** can only contain definitions for the CPU application class. If it contains definitions for other application classes, the definitions are ignored. Additionally, if a file is removed, then the definition for the application class is removed.

## Defining Which Override Files to Use

The */AgentSetup/ExternalOverride* configuration variable determines which file(s) to use.

|                                |  |
|--------------------------------|--|
| <b>Format and Type of Data</b> | Text string indicating a file name. The string can contain macro variables.  |
| <b>Default Value</b>           | (empty string = no file)   |
| <b>Minimum and Maximum</b>     | not applicable   |
| <b>Dependencies</b>            | <i>/AgentSetup/ExternalOverridePolltime</i> specifies how often the file is checked for changes.                         |
| <b>Recommendations</b>         | Use macro variables to specify different names for different situations and name the external override files accordingly |

## Using Macro Variables in Override Files

You can specify different files for different situations by using macro variables.

---

### Example

---

To use different files during the day and night, you must use a macro variable and use it in the configuration variable.

```
/AgentSetup/ExternalOverride = "% {hostname}_% {timeofday}
```

If /timeofday is set to “day”, the PATROL Agent uses the file  
%PATROL\_HOME%/lib/admin/myhost\_day

If /timeofday is set to “night”, the PATROL Agent uses the file  
%PATROL\_HOME%/lib/admin/myhost\_night

---

## External Override File Poll Cycle

The */AgentSetup/ExternalOverridePolltime* configuration variable determines how often the PATROL Agent checks the external file for changes.

|                                |   |
|--------------------------------|---|
| <b>Format and Type of Data</b> | integer, seconds<br>0—external file polling is disabled<br>60+—the polling cycle in seconds |
| <b>Default Value</b>           | [TBD]   |
| <b>Minimum and Maximum</b>     | 0 or 60+, no maximum  |
| <b>Dependencies</b>            | <i>/AgentSetup/ExternalOverride</i> specifies which file is checked for changes.            |
| <b>Recommendations</b>         | none  |

To manually check if the file has changed, use the  
%OVERRIDE CHECK command.

# External Override File Format

The file format is the standard ini file format. The name of a section identifies the parameter to be overridden. The attributes for a section indicate the properties and values to be used. All lines starting with a # are treated as comments and ignored.

## Sections

The name of each section identifies the parameter to be overridden. To override a parameter for a specific application instance, use the following format for the section name:

```
[/APPLICATION/INSTANCE_SID/PARAMETER]
```

To override a parameter for all instances of an application class, leave the INSTANCE\_SID blank:

```
[/APPLICATION/ /PARAMETER]
```

## Attributes

The attribute and value pairs for each section identify the new values to be used. Table 17-1 lists the available attribute and value pairs.

**Table 17-1 Attribute and Value Pairs**

| Attribute  | Description   |
|--|---|
| ACTIVE   | Boolean. This attribute defines whether the parameter is active or not.   |
| INTERVAL   | Integer. This integer attribute defines the number of seconds for the polling interval. This attribute does not apply to consumer parameters. |
| BORDER_ACTIVE<br>ALARM1_ACTIVE<br>ALARM2_ACTIVE    | Boolean. These attributes define whether the alarm ranges are active or not for the border, alarm1, and alarm2 ranges.                        |
| BORDER_MINIMUM<br>ALARM1_MINIMUM<br>ALARM2_MINIMUM | Integer. These attributes define the minimum values for the border, alarm1, and alarm2 ranges.  |

**Table 17-1 Attribute and Value Pairs**

| <b>Attribute</b>  | <b>Description</b>   |
|---|--|
| BORDER_MAXIMUM<br>ALARM1_MAXIMUM<br>ALARM2_MAXIMUM                | Integer. These attributes define the maximum values for the border, alarm1, and alarm2 ranges.   |
| BORDER_STATE<br>ALARM1_STATE<br>ALARM2_STATE                      | Constant. These attributes define the state for the border, alarm1, and alarm2 ranges. Possible values are OK, WARN, WARNING, and ALARM.   |
| BORDER_ALARM_WHEN<br>ALARM1_ALARM_WHEN<br>ALARM2_ALARM_WHEN       | Constant. These attributes define when the alarms occur for the border, alarm1, and alarm2 ranges. Possible values are ALARM_INSTANT, ALARM_AFTER_N, and ALARM_AFTER_RECOVERY.           |
| BORDER_ALARM_WHEN_N<br>ALARM1_ALARM_WHEN_N<br>ALARM2_ALARM_WHEN_N | Integer. These attributes define the number of consecutive times the parameter must have a value within the range before the alarm occurs. This attribute is only ALARM_AFTER_N is used. |
| BORDER_DO_RECOVERY<br>ALARM1_DO_RECOVERY<br>ALARM2_DO_RECOVERY    | Boolean. These attributes define whether the recovery action is executed or not.   |

**External File Example**


---

```

# The values are chosen for demo purposes only
# Do not use these in a production environment !

# Override all instances of LDldDiskTimePercent

[/NT_LOGICAL_DISKS//LDldDiskTimePercent]

ACTIVE=1
BORDER_ACTIVE=1
BORDER_MINIMUM=10
BORDER_MAXIMUM=90
BORDER_STATE=ALARM
BORDER_ALARM_WHEN=ALARM_INSTANT
BORDER_ALARM_WHEN_N=0
ALARM1_ACTIVE=1
ALARM1_MINIMUM=50
ALARM1_MAXIMUM=70

```

```
ALARM1_STATE=WARN
ALARM1_ALARM_WHEN=ALARM_INSTANT
ALARM1_ALARM_WHEN_N=0
ALARM2_ACTIVE=1
ALARM2_MINIMUM=70
ALARM2_MAXIMUM=90
ALARM2_STATE=ALARM
ALARM2_ALARM_WHEN=ALARM_INSTANT
ALARM2_ALARM_WHEN_N=0
ALARM2_DO_RECOVERY=NO

# Override PAWorkRateExecsMin for only the
# PATROL_NT application instances

[/PATROL_NT/PATROL_NT/PAWorkRateExecsMin]

ACTIVE=1
BORDER_ACTIVE=1
BORDER_MINIMUM=0
BORDER_MAXIMUM=100
BORDER_STATE=ALARM
BORDER_ALARM_WHEN=ALARM_INSTANT
BORDER_ALARM_WHEN_N=0
ALARM1_ACTIVE=1
ALARM1_MINIMUM=30
ALARM1_MAXIMUM=60
ALARM1_STATE=WARN
ALARM1_ALARM_WHEN=ALARM_INSTANT
ALARM1_ALARM_WHEN_N=0
ALARM2_ACTIVE=1
ALARM2_MINIMUM=60
ALARM2_MAXIMUM=100
ALARM2_STATE=ALARM
ALARM2_ALARM_WHEN=ALARM_INSTANT
ALARM2_ALARM_WHEN_N=0
```

---

## Deleting the External Override File

If the PATROL Agent cannot locate the external override file because it is moved or deleted, the PATROL Agent considers the file unchanged. This makes it easy for you to implement a software distribution mechanism that moves the existing external override file to a backup location before copying over the new external override file.

In order to clear all overrides from an external override file, you must delete the contents of the file and leave the blank file in place until the next polling cycle.

## Additional Functions

The PATROL Agent also includes the following commands that you can use with external override files:

- `%OVERRIDE DUMP`
- `%OVERRIDE REAPPLY`
- `%OVERRIDE CHECK`

### **%OVERRIDE DUMP**

This command outputs the following information:

- the information in the external file
- the polling interval
- the name of the current external override file

### **%OVERRIDE REAPPLY**

This command reapplies all of the settings in the external override file. This command is only useful when PSL function overrides are also used. When this command is run, all overrides in the external override file are applied, replacing any corresponding overrides created by the PSL function.

## **%OVERRIDE CHECK**

This command checks if the external file has changed.

---

# List of PATROL Agent Variables

This appendix lists variables that affect the PATROL Agent. It is organized by variable type. The appendix consists of a series of tables. Each table lists the variable by name, describes its purpose and format, and references where more information about the variable can be found. This appendix contains the following sections:

|                                    |      |
|------------------------------------|------|
| Configuration Variables . . . . .  | A-2  |
| Agent Setup . . . . .              | A-2  |
| Agent Tuning . . . . .             | A-8  |
| Security . . . . .                 | A-9  |
| SNMP. . . . .                      | A-10 |
| Environment Variables . . . . .    | A-13 |
| Components and Platforms . . . . . | A-13 |
| List of Variables . . . . .        | A-15 |

# Configuration Variables

This section lists the Agent Configuration Variables in alphabetic order and provides a description of the variable, the format of its value, and a reference to the documentation that describes it in detail.

## Agent Setup

In the PATROL Agent Configuration Utility, these variables are located in `/AgentSetup`.

**Table A-1** `/AgentSetup/` Variables (Part 1 of 6)

| Variable                            | Description   | Format                       | For Details, See Page |
|-------------------------------------|---|------------------------------|-----------------------|
| <code>_name_</code>                 | PATROL Agent Setup  | text string (without spaces) | cannot be changed     |
| <code>_trailing_comma_</code>       | Need for internal operations  | text string (without spaces) | cannot be changed     |
| <code>_type_</code>                 | PATROL Agent Setup type   | text string (without spaces) | cannot be changed     |
| <code>accessControlList</code>      | determines which users can access the Agent from which hosts in which connection mode   | user/host/mode               | 5-9                   |
| <code>application.filterList</code> | specifies a set of application instances that you want to include or exclude from monitoring, depending upon the type of filter | text string (no spaces)      | 7-18                  |
| <code>application.filterType</code> | determines the type of filter for a corresponding filter list<br><br>You must create this variable.                             | text string (no spaces)      | 7-19                  |
| <code>appl.OSdefaultAccount</code>  | specifies an account under which the Agent runs all parameter and recovery actions for the application                          | text string (no spaces)      | 4-3                   |

**Table A-1 /AgentSetup/ Variables (Part 2 of 6)**

| <b>Variable</b>  | <b>Description</b>   | <b>Format</b>            | <b>For Details, See Page</b> |
|--|--|--------------------------|------------------------------|
| <i>appl.instName.OSdefault</i><br>Account              | specifies an account under which the Agent runs all parameter and recovery actions for the application instance  | text string (no spaces)  | 4-4                          |
| <i>appl.instName.OSdefault</i><br>AccountAppliesToCmds | determines whether commands issued from the console run under the console log-in account or the <i>appl.OSdefaultAccount</i>   | boolean (yes, no)        | 4-5                          |
| BindToAddress  | specifies which IP Address the PATROL Agent uses to communicate in a system with multiple network cards  | numeric                  | 5-5                          |
| comSecurityLevel                                       | controls the level of information provided by the PATROL Agent to the DCOM client  | alphabetic range         | C-3                          |
| comUserAllowed   | controls who can access the PATROL Agent DCOM server based upon group names and user names   | text string              | C-3                          |
| defaultAccount   | specifies the user account under which the PATROL Agent runs all parameters, recovery actions, and application discovery procedures when an account is not explicitly specified<br><br>It must be in administrative group. | text string (no spaces)  | 4-2                          |
| defaultAccountShell                                    | allows you to specify which shell the PATROL Agent uses for the process spawned by the PATROL Agent default account.   | text string (shell name) | 4-2                          |

**Table A-1 /AgentSetup/ Variables (Part 3 of 6)**

| <b>Variable</b>          | <b>Description</b>   | <b>Format</b>   | <b>For Details,<br/>See Page</b> |
|--------------------------|--|---|----------------------------------|
| defaultDisplay           | controls where Unix applications display their output  | text string (no spaces)   | 5-3                              |
| disabledKMs              | prevents Knowledge Modules from being loaded for all operating systems and machine types specified                           | text string (comma-separated, item list)                                    | 7-10                             |
| disabledKMsArch          | prevents Knowledge Modules from being loaded for a particular operating system and machine type                              | text string (comma-separated, item list)                                    | 7-11                             |
| EnableSysOutputAclCheck  | allows you to use the S option in an ACL to restrict or allow display of the system output window and command task creation. | boolean (0/1)<br>disable=0<br>enable=1                                      | 5-13                             |
| ExternalOverride         | determines which file to use for parameter override values.  | Text string indicating a file name. The string can contain macro variables. | 17-11                            |
| ExternalOverridePolltime | determines how often the PATROL Agent checks the external file for parameter override changes.                               | integer, seconds  | 17-13                            |
| historyRetentionPeriod   | determines number of days that collected parameter values are kept   | numeric (days)  | 12-8                             |
| IPAddresses              | lists all the possible IP addresses through which the Agent can communicate  | numeric (IP Address)  | 5-7                              |
| localPortForRemoteOpen   | sets a local UDP port-number for the agent to run the PSL remote_open() function   | numeric (port number)   | 4-9                              |
| maxAgentMessageLimit     | determines the number of messages written to the PATROL Agent error log  | numeric   | 11-18                            |

**Table A-1 /AgentSetup/ Variables (Part 4 of 6)**

| <b>Variable</b>     | <b>Description</b>   | <b>Format</b>   | <b>For Details, See Page</b> |
|---------------------|--|---|------------------------------|
| maxProcessLimit     | limits the number of processes that can run simultaneously                     | numeric   | H-20                         |
| pemCacheSize        | sets the size of the cache used by the agent for event management              | numeric   | 11-6                         |
| pemEvMemRetention   | sets the number of events the PEM engine keeps in memory for each object       | numeric   | 11-7                         |
| pemIssueV30traps    | determines whether PATROL uses PATROL Version 3.0 formats to issue SNMP traps  | boolean (yes, no)   | 15-29                        |
| pemIssueV31traps    | determines whether PATROL uses PATROL Version 3.1 formats to issue SNMP traps  | boolean (yes, no)   | 15-29                        |
| pemLogName          | sets the name of the event log repository                                      | text string (no spaces)                                   | 11-5                         |
| pemLogSize          | sets the maximum size of the event log   | numeric   | 11-5                         |
| pemPFSnmpPattern    | establishes the pattern you want to filter for in the description of the event | text string   | 15-25                        |
| pemPFSnmpStatusMask | sets the event statuses you want to filter                                     | alphabetic range  | 15-28                        |
| pemPFSnmpTypeMask   | sets the event types you want to filter  | alphabetic range  | 15-27                        |
| pemPFSnmpEidRange   | sets the range of event IDs you want to filter                                 | numeric   | 15-24                        |
| pemPFSnmpEndTime    | sets the end time  | MMDDhhmm[yy]<br>(date, time, and year, which is optional) | 15-25                        |
| pemPFSnmpEvClass    | sets the event class you want to filter  | text string (event class name)                            | 15-24                        |

**Table A-1 /AgentSetup/ Variables (Part 5 of 6)**

| <b>Variable</b>     | <b>Description</b>  | <b>Format</b>                            | <b>For Details, See Page</b>              |
|---------------------|---|--|---|
| pemPFSnmpNSeverity  | sets the severity level that triggers SNMP traps  | numeric (range 0-5)                      | 15-22                                     |
| pemPFSnmpNode       | specifies the node where the event occurred   | text string                              | This variable is reserved for future use. |
| pemPFSnmpOrigin     | specifies the application where the event occurred  | text string                              | 15-23                                     |
| pemPFSnmpStartTime  | sets the start time   | MMDDhhmm[yy]                             | 15-25                                     |
| pemSnmpSupport      | determines whether PEM triggers SNMP events   | boolean (yes, no)                        | 15-21                                     |
| PerfGetDebugFile    | specifies debug file for the persistent data collection program   | text string                              | H-23                                      |
| PerfGetMaxTimes     | determines the number of data points collected by the persistent data collection program before it restarts | numeric                                  | H-23                                      |
| PerfGetTimeout      | determines how long the Agent waits for a response from the persistent data collection program              | numeric (milliseconds)                   | H-22                                      |
| PerfMaxRestartCount | limits the number of patrolperf instances the PATROL Agent can spawn.                                       | numeric (number of processes)            | H-20                                      |
| PortConnectType     | specifies which network card (NIC) the PATROL Agent uses to communicate                                     | numeric                                  | 5-5                                       |
| preloadedKMs        | specifies which Knowledge Modules are preloaded for all operating systems and machine types                 | text string (comma-separated, item list) | 7-7                                       |
| preloadedKMsArch    | specifies which Knowledge Modules are preloaded for particular operating system and machine types           | text string (comma-separated, item list) | 7-8                                       |

**Table A-1 /AgentSetup/ Variables (Part 6 of 6)**

| <b>Variable</b>             | <b>Description</b>   | <b>Format</b>                             | <b>For Details, See Page</b> |
|-----------------------------|--|---|------------------------------|
| prmHistCacheFlushTimer      | sets the period of time (interval at which) that the cache is flushed to the history database                  | numeric                                   | 12-7                         |
| prmHistCacheMaxFlushTime    | defines how much time the agent spends writing cache data to a history file.                                   | numeric (seconds)                         | 12-8                         |
| prmHistCacheSize            | sets the number of data points kept in cache   | numeric                                   | 12-7                         |
| PsIDebug                    | sets the run-time error level for PSL  | numeric                                   | H-21                         |
| sessionsInitMayFail         | determines whether the PATROL Agent will run if no sessions to the PATROL Console can be established           | boolean (yes, no)                         | 5-4                          |
| snmpConfigAccess            | determines whether SNMP support can write to the configuration database  | boolean (0/1)                             | 16-9                         |
| staticApplication           | specifies a list of static applications. The applications are marked as static if they are dynamically loaded. | text string (comma-separated, item list)  | 7-9                          |
| suppressConsoleInfoMsgMask  | controls whether PEM messages are logged in the PEM Log and are displayed in the PEM Console                   | alpha character (ACDOP)                   | 5-2                          |
| suppressSystemOutputMsgMask | suppresses or allows the display of KM-related messages in the system output window.                           | boolean (yes, no)<br>Default value is no. | 5-3                          |
| timeZone                    | allows you to define time zones that do not exist in the PATROL Agent time zone table.                         | TimeZone/Offset<br>Value                  | 4-4                          |

# Agent Tuning

In the PATROL Agent Configuration Utility, these variables are located in /AgentSetup/AgentTuning.

**Table A-2 /AgentSetup/AgentTuning Variables (Part 1 of 2)**

| <b>Variable</b>        | <b>Description</b>  | <b>Format</b> | <b>For Details, See Page</b> |
|------------------------|---|---------------|------------------------------|
| agentPriority          | sets the operating system scheduling priority (referred to as the nice value in Unix) for the PATROL Agent process  | numeric       | H-12                         |
| applCheckCycle         | sets the application discovery rate in seconds  | numeric       | H-9                          |
| getProcsCycle          | sets process cache refresh rate in seconds  | numeric       | H-9                          |
| procCachePriority      | sets the operating system scheduling priority (referred to as the nice value in Unix) for the process that refreshes the process cache                    | numeric       | H-13                         |
| procCacheSchedPriority | sets the internal scheduling priority for the process that refreshes the process cache  | numeric       | H-13                         |
| pslInstructionMax      | sets the maximum number of instructions within pslInstructionPeriod seconds that a PSL process can execute without incurring an internal scheduling delay | numeric       | H-6                          |
| pslInstructionPeriod   | sets period of time in seconds that a PSL process can execute the maximum number of instructions without incurring an internal scheduling delay           | numeric       | H-7                          |
| runqDelta              | sets gap (in seconds) between processes in the PATROL Agent's run queue   | numeric       | H-17                         |

**Table A-2 /AgentSetup/AgentTuning Variables (Part 2 of 2)**

| <b>Variable</b>    | <b>Description</b>   | <b>Format</b> | <b>For Details, See Page</b> |
|--------------------|--|---------------|------------------------------|
| runqDeltaIncrement | sets the increment (in seconds) used when checking for a gap in the PATROL Agent's run queue     | numeric       | H-17                         |
| runqMaxDelta       | sets the maximum delay (in seconds) for a process  | numeric       | H-18                         |
| runqSchedPolicy    | sets PATROL Agent's scheduling policy  | numeric       | H-15                         |
| userPriority       | sets operating system scheduling priority (the nice value) for processes created by PATROL Agent | numeric       | H-12                         |

## Security

In the PATROL Agent Configuration Utility, these variables are located in /AgentSetup/security.

**Table A-3 /AgentSetup/security Variables**

| <b>Variable</b>         | <b>Description</b>                     | <b>Format</b>     | <b>For Details, See Page</b> |
|-------------------------|--|-------------------|------------------------------|
| ExtendedSecurityEnabled | enables the PATROL Agent's ESI support | boolean (yes, no) | 13-10                        |

# SNMP

In the PATROL Agent Configuration Utility, these variables are located in /snmp.

**Table A-4 /SNMP Variables (Part 1 of 3)**

| <b>Variable</b>     | <b>Description</b>   | <b>Format</b>                 | <b>For Details, See Page</b> |
|---------------------|--|-------------------------------|------------------------------|
| _name_              | the SNMP name  | text string (snmp)            | cannot be changed            |
| _type_              | the SNMP type  | text string (DATA_REPOSITORY) | cannot be changed            |
| accessControlList   | controls access to the agent based on SNMP host name. The SNMP ACL does not use username or mode   | */hostname                    | 15-4                         |
| agent_auto_start    | determines whether the SNMP Agent support (SNMP sub-agent) is started when the PATROL Agent starts | boolean (yes, no)             | 15-9                         |
| agent_r_community   | sets the read community string for PATROL SNMP Agent (PATROL SNMP Master Agent) operations         | text string                   | 15-11                        |
| agent_w_community   | sets the write community string for PATROL SNMP Agent (PATROL SNMP Master Agent) operations        | text string                   | 15-12                        |
| default_retries     | sets the number of retries for PSL and SNMP operations   | numeric                       | 16-7                         |
| default_port        | sets the default port number which the PATROL Agent uses to open sessions with SNMP agents         | numeric                       | 16-7                         |
| default_r_community | sets the default community name for SNMP get and getnext operations in PSL                         | text string                   | 16-5                         |
| default_timeout     | sets the timeout value in milliseconds for PSL and SNMP operations                                 | numeric (milliseconds)        | 16-6                         |

**Table A-4 /SNMP Variables (Part 2 of 3)**

| <b>Variable</b>       | <b>Description</b>   | <b>Format</b>     | <b>For Details,<br/>See Page</b>          |
|-----------------------|--|-------------------|---|
| default_w_community   | sets default community name for SNMP set operations in PSL                                   | text string       | 16-6                                      |
| ignoreIPAddress       | Ignore IP Address of responses to SNMP "get" requests.                                       | boolean (yes, no) | 15-14                                     |
| master_agent_port     | sets the default listening port  | numeric           | This variable is reserved for future use. |
| masterAgentConfigDir  | points to the directory containing the PATROL SNMP Master Agent configuration file           | text string       | 15-6                                      |
| masterAgentConfigName | contains the name of PATROL SNMP Master Agent configuration file                             | text string       | 15-6                                      |
| masterAgentDir        | points to the directory containing the PATROL SNMP Master Agent executable file              | text string       | 15-5                                      |
| masterAgentName       | contains the name of the PATROL SNMP Master Agent executable file                            | text string       | 15-5                                      |
| masterAgentParamDir   | points to the directory containing the PATROL SNMP Master Agent nonvolatile information file | text string       | 15-7                                      |
| masterAgentParamName  | contains the name of PATROL SNMP Master Agent nonvolatile information file                   | text string       | 15-7                                      |
| masterAgentStartLine  | sets the command that is used on a Unix system to start the PATROL SNMP Master Agent         | text string       | 15-4                                      |

**Table A-4 /SNMP Variables (Part 3 of 3)**

| <b>Variable</b>       | <b>Description</b>   | <b>Format</b>   | <b>For Details,<br/>See Page</b> |
|-----------------------|--|---|----------------------------------|
| masterAgentWorkingDir | points to the working directory for the PATROL SNMP Master Agent (contains the PATROL SNMP Master Agent executable file) on Unix | text string   | 15-8                             |
| mibFileName           | contains the MIB file that the PATROL Agent loads for PSL SNMP management functions  | text string   |                                  |
| piV1m_list            | contains the list of SNMPV1 managers that are interested in getting automatic SNMP traps from the PATROL Agent                   | text string<br>(host/port1/access,<br>host2/port2/access) | 15-20                            |
| support               | determines whether SNMP support is available   | boolean (yes, no)   | 16-5                             |
| sysContact            | contains the value of MIB-II system.sysContact   | text string   | 15-13                            |
| sysLocation           | contains the value of MIB-II system.sysLocation  | text string   | 15-13                            |
| sysName               | contains the value of MIB-II system.sysName  | text string   | 15-13                            |
| trap_port             | sets the UDP port number for SNMP trap listening   | numeric   | 15-6                             |
| trapConfTable         | determines whether to issue SNMP traps to managers of the pre-configured list in the PATROL SNMP Master Agent configuration file | boolean (yes, no)   | 15-13                            |
| trapMibTable          | determines whether to issue SNMP traps to managers of the pre-configured list in the PATROL SNMP Master Agent configuration file | boolean (yes, no)   | 15-14                            |

# XPC

In the PATROL Agent Configuration Utility, these variables are located in AgentSetup/XPC.

**Table A-5** /AgentSetup/XPC Variables

| Variable                                    | Description                                | Format  | For Details, See Page |
|---|--|---|-----------------------|
| <b>&lt;xpcserver&gt;.xpc_defaultAccount</b> | specifies a default account for xpc server | text string<br><xpcserver>.xpc_defaultAccount | 4-5                   |

**Note**

This variable is available only for Microsoft Windows platforms.

## Environment Variables

PATROL uses environment variables to control the environment in which the PATROL Agent and PATROL Console run. This section lists all PATROL environment variables, describes the purpose of each, provides a default value, and indicates which PATROL components and which platforms the variable affects.

## Components and Platforms

Environment variables can affect one or both of the PATROL components. Most of the variables created for PATROL run on all platforms, but not all variables do.

For each variable, the C\P column in Table A-6 lists which components the variables affect and on which platforms they are valid. The values are

- A** PATROL Agent only
- B** Both PATROL Console and PATROL Agent

|            |                            |
|------------|----------------------------|
| <b>C</b>   | PATROL Console only        |
| <b>C7</b>  | PATROL 7.x consoles only   |
| <b>CS</b>  | PATROL Console Server only |
| <b>U</b>   | Unix                       |
| <b>W</b>   | Windows                    |
| <b>VMS</b> | OpenVMS                    |
| <b>ALL</b> | All platforms              |

## List of Variables

Table A-6 lists alphabetically environment variables that affect the agent, the console, or both. In the Default column, the table uses Unix naming conventions: a dollar sign to indicate a variable and a forward slash (/) to delimit directories. These default values, after making the appropriate platform substitutions, apply to all platforms listed in the C\P column for the respective environment variable.

**Table A-6 Environment Variables Used by PATROL Console and PATROL Agent (Part 1 of 4)**

| Environment Variable  | Default                  | Description   | C\P     |
|-----------------------|--------------------------|---|---------|
| BMC_ROOT              | /opt/bmc                 | Top level directory for all BMC products.                               | ALL     |
| COMSPEC               | c:\command.com           | COMMAND Interpreter used for Windows                                    | W       |
| DUMP_CORE_ON_XT_ERROR |                          | Dump core due to malloc errors  |         |
| ETC_HOSTS_SCRIPT      | NULL                     |   |         |
| GETHOSTENT_WORKAROUND | NULL                     |   |         |
| HOME                  | \$HOME                   | Environment variable where the PATROL Console writes its customizations | U       |
| PATROL_ADMIN          | patrol                   | Set to user that is to own log files that the Patrol Agent writes       | A, ALL  |
| PATROL_ARCHIVE        | \$PATROL_HOME/./archives | TOC files stored here   | ALL     |
| PATROL_BIN            | \$PATROL_HOME/bin        | Location of the PATROL binary files                                     | B, ALL  |
| PATROL_CACHE          | \$HOME/patrol            | Console directory for KM customizations                                 | C, U, W |
| PATROL_CFG            | \$HOME/patrol/config     | Console directory where xpconfig stores Agent configuration files       | ALL     |
| PATROL_CONFIG         | \$PATROL_HOME/config     | Directory where the Agent configuration database is stored              | A, ALL  |

**Table A-6 Environment Variables Used by PATROL Console and PATROL Agent (Part 2 of 4)**

| <b>Environment Variable</b> | <b>Default</b>                        | <b>Description</b>   | <b>C/P</b> |
|-----------------------------|---------------------------------------|--|------------|
| PATROL_CONFIG_PORT          | none                                  | See Chapter 6, “Support of Clusters and Failovers”   | A, ALL     |
| PATROL_DEBUG                | none                                  | Variable that sets the debug level when the <code>-debug</code> option is used and a level is not supplied | ALL        |
| PATROL_DESKTOP              | \$HOME/patrol/desktop                 | Local directory where desktop configurations are stored  | C, U, W    |
| PATROL_GLOBAL_LIB           | \$PATROL_HOME/lib                     | Global PATROL library directory  | B, ALL     |
| PATROL_HELP                 | \$PATROL_GLOBAL_LIB/app-defaults/help | Directory containing help files  | C, U, W    |
| PATROL_HISTORY              | \$PATROL_LOG/history                  | Location of parameter history data files   | A, ALL     |
| PATROL_HISTORY_PORT         | none                                  | See Chapter 6, “Support of Clusters and Failovers”   | A, ALL     |
| PATROL_HOME                 | \$PATROL_HOME/platform                | Main install directory for PATROL platform-specific files  | B, ALL     |
| PATROL_KM                   | \$PATROL_GLOBAL_LIB/knowledge         | Global Knowledge Module directory  | B, ALL     |
| PATROL_LICENSE              | \$PATROL_GLOBAL_LIB                   | Location of the PATROL License file  |            |
| PATROL_LOCAL_CHART          | \$PATROL_CACHE/chart                  | Location where the Chart Utility stores preferences  |            |
| PATROL_LOCAL_KM             | \$PATROL_CACHE/knowledge              | Local Knowledge Module directory   | C          |
| PATROL_LOCAL_PSL_APPS       | \$PATROL_CACHE/psl                    | Local PSL directory  | C          |
| PATROL_LOG                  | \$PATROL_HOME/log                     | Patrol Agent log directory   | A          |
| PATROL_MACHINE_TYPE         | ‘platform’                            | Override for the Patrol Machine Type response. Useful on SVR4 machines that need a different icon.         | U          |

**Table A-6 Environment Variables Used by PATROL Console and PATROL Agent (Part 3 of 4)**

| <b>Environment Variable</b> | <b>Default</b>             | <b>Description</b>  | <b>C/P</b>  |
|-----------------------------|----------------------------|---|-------------|
| PATROL_MAXLOG               | 5                          | limits the number of error log backup files that can be generated. The default value is five backup files. The maximum value that can be set is 50, and minimum is 1. | A           |
| PATROL_MAX_FILE_DESCRIPTORS | 1024                       | See "Limit the Number of Files Opened Simultaneously" on page H-19.   | B, ALL      |
| PATROL_MIBFILE              | none                       | MIB filename for SNMP support (used only if not set in Agent's configuration).  |             |
| PATROL_PORT                 | 3181                       | See "Setting the Default Port Number on Unix Only" on page 4-8  | A, U        |
| PATROL_PSL_APPS             | \$PATROL_GLOBAL_LIB/psl    | Global PSL directory  | B, ALL      |
| PATROL_QRY                  | \$HOME/query               | Local directory where query results are stored from Agent Query   | C, U, W     |
| PATROL_REMOTE               | \$PATROL_HOME/remote       | Directory where received remote file transfers are written  | A, ALL      |
| PATROL_ROOT                 | /OPT/BMC/PATROL7           | Top level directory for PATROL 7 products   | C7, CS, ALL |
| PATROL_SOUNDS               | \$PATROL_GLOBAL_LIB/sounds | Location of PATROL Sounds directory   | C, U, W     |
| PATROL_TMP                  | \$PATROL_CACHE/tmp         | Override for \$TMP  | C, U, W     |
| PATROL_VIRTUALNAME          | none                       | cluster-specific variable for history and configuration   | 6-28, 6-29  |
| PATROL_VIRTUALNAME_PORT     | none                       | cluster-specific variable for history and configuration   | 6-28, 6-29  |
| RTHOME                      |                            | RTHOME is set when you install an RTserver and has the following default value:<br>installationdirectory\common\smartsockets\etc\ss                                   | A, CS       |

**Table A-6 Environment Variables Used by PATROL Console and PATROL Agent (Part 4 of 4)**

| <b>Environment Variable</b> | <b>Default</b>                             | <b>Description</b>   | <b>C/P</b> |
|-----------------------------|--|--|------------|
| RTSERVERS                   |  | The RTSERVERS environment variable tells the agent which RTserver(s) to connect to   | A          |
| SPEAKER                     | \$PATROL_HOME/bin/player                   | Executable location to play sounds   |            |
| TMP                         | \$TMP                                      | Override for /tmp  |            |
| TMP_PATROL                  | /tmp/patrol                                | On NFS mounted file systems, this variable must be set to a LOCAL file system to ensure that the PEM lock file will be properly created. | B, U       |
| XBMLANGPATH                 | \$PATROL_GLOBAL_LIB/images                 | Images sub-directory   | C, U       |
| XKEYSYMDB                   | \$PATROL_GLOBAL_LIB/app-defaults/XkeysymDB | File with X defaults   | C, U       |
| YPCAT_HOSTS_SCRIPT          | NULL                                       |  |            |

### **RTSERVERS Environment Variable**

The RTSERVERS environment variable tells the agent which RTserver(s) to connect to, for example: tcp:nebula:2059.

The agent connects to the first RTserver in the list. If the connection breaks, the agent connects to the next RTserver in the list.

You must set the RTSERVERS environment variable in either of the following situations:

- You want to view PATROL Agent data from a PATROL 7.x console.
- There is no RTserver installed on a computer in the same subnet.
- The RTserver is not using the default port number of 2059.

The PATROL Console Server, PATROL Agent, and PATROL 7.x consoles can detect RTservers within their own subnets, but they cannot detect RTservers outside of their own subnets. If you have installed one of those components in a subnet that has no RTserver, the only way it can communicate with an RTserver is if you define the RTSERVERS environment variable.

The format of the RTSERVERS environment variable is as follows:

*tcp: computer\_name: port\_number*

where *computer\_name* is the name of a computer running an RTserver (a member of the RTserver cloud), and *port\_number* is the port number that it is using to broadcast.



---

# Changing Variables in a File Manually

This appendix describes how to change variables manually rather than using pconfig, wpconfig, or xpconfig utilities. When changing variables manually, take care not to create syntax errors that are easily avoided with the utilities. The types of changes that this appendix describes are made to files with cfg extensions. Changes to these files have no effect unless they are applied to specific agents using one of the utilities: pconfig, wpconfig, and xpconfig.

This appendix contains the following sections:

|   |     |
|---|-----|
| Overview of Changing Variables in a File Manually . . . . . | B-2 |
| File Names to Use for Change Files . . . . .                | B-2 |
| Parts of a Change File . . . . .                            | B-2 |
| Editing the Change File Using a Text Editor . . . . .       | B-4 |

# Overview of Changing Variables in a File Manually

You can change variables manually by creating or editing a change file. A change file is an ASCII file that lists each variable whose value (or value list) you want to change. The entry for each variable specifies whether to append, delete, merge, or replace one or more values. For information on editing a change file, see the section “Editing the Change File Using a Text Editor” on page B-4.

## File Names to Use for Change Files

You should name change files *hostname.cfg*, where *hostname* is the name of the machine where the PATROL Agent is running.

## Parts of a Change File

A change file should have the general format shown in Figure B-1.

**Figure B-1 PATROL Agent Configuration Change File Format**

---

```
PATROL_CONFIG
"VARIABLE"={
    APPEND ="VALUE[ ,VALUE... ]" ,
    DELETE[="VALUE[ ,VALUE... ]" , ]
    MERGE  ="VALUE[ ,VALUE... ]" ,
    REPLACE="VALUE[ ,VALUE... ]"
}
...
"/Agentsetup/preloadedKMs"={
    APPEND = "UNIX_2.1.km,ORACLE6.km" ,
    DELETE = "SYBASE.km" ,
    MERGE  = "INGRES.km,ORACLE7.km"
}
"AgentSetup/___type___"={
    REPLACE = "DATA_REPOSITORY"
}
}
```

Command necessary for utility program to recognize the file

Generic Example

---

## Editing the Change File Using a Text Editor

---

**Summary:** If you want to change the values for variables manually, you must edit the appropriate change file using a text editor.

---

### Before You Begin

Before you begin editing the change file with a text editor, make sure you have done the following:

- You have reviewed and understood Chapter 3, “Background About Configuring the PATROL Agent.”
- You know the name of the change file you want to edit.
- You have write permission on the change file.

When you edit the configuration change file, you should observe these guidelines:

- The `PATROL_CONFIG` command must be before the changes.
- A comma must end every entry except the last one.
- The change entries can be in any order and are applied in the order of appearance.
- White space is ignored except within double quotation marks.

### To Edit the Change File

- Step 1** Access the text editor you want to use.
- Step 2** Edit the file to make the changes to the appropriate variables.
- Step 3** Save the file.

---

# Configuring a DCOM Interface to the PATROL Agent

This appendix describes how to set up the PATROL Agent for use with the Distributed Component Object Model (DCOM) programming interface. The DCOM interface (a tool for Windows) provides developers of Component Object Model (COM) compatible applications access to data collected by one or more PATROL Agents.

This appendix contains the following sections:

|  |     |
|--|-----|
| Tasks to Configure a DCOM Interface to the PATROL Agent . . . . .                  | C-2 |
| Setting PATROL Agent Configuration Variables. . . . .                              | C-3 |
| Configuring DCOM on PATROL Agents . . . . .  | C-4 |
| Configuring Your Computer for DCOM Communication with<br>the PATROL Agent. . . . . | C-6 |

For additional information about programming using the COM and DCOM protocols, refer to the Microsoft World Wide Web site at <http://www.microsoft.com>.

---

### Note

---

The DCOM interface is only available on Windows.

---

# Tasks to Configure a DCOM Interface to the PATROL Agent

Before you can use your application (DCOM client) to access data on one or more PATROL Agents (DCOM servers), you need to perform one or both of the following tasks:

- configure the computers hosting PATROL Agents (DCOM servers)

You can do this either by setting the PATROL Agent COM configuration variables *or* by running the Microsoft DCOM configuration utility.

- configure the computers hosting your application (DCOM clients) by running the PATROL Client Configuration Utility

## Setting PATROL Agent Configuration Variables

---

**Summary:** In this task you will assign values to the PATROL Agent configuration variables that will allow a DCOM client to communicate with the agent.

---

**Step 1** Launch wpconfig for the PATROL Agent you are configuring.

**Step 2** Assign values to the PATROL Agent Setup variables described in Table C-1.

**Table C-1 PATROL Agent COM Variables**

| Variable         | Values  | Description   |
|------------------|---|---|
| comSecurityLevel | W (default), N, R, F                            | Controls the level of information provided by the PATROL Agent to the DCOM client.<br><b>W</b> —DCOM client can receive and change data on the agent<br><b>N</b> —No DCOM interface allowed<br><b>R</b> —DCOM client can receive data from the agent.<br><b>F</b> —DCOM client can receive, change, and use PsExecute() to run OS commands.   |
| comUserAllowed   | Empty String (default), Group Names, User Names | Controls who can access the PATROL Agent DCOM server. Listed users, members of groups, and the user who started the agent may access the agent DCOM server. Separate multiple values with the “ ” character (no spaces). This variable, supported only by Windows NT 4.0 and later, takes precedence over any security set by the Microsoft DCOM Configuration Utility (see “Configuring DCOM on PATROL Agents” on page C-4). If you set comUserAllowed to the empty string, security is determined by the DCOM Configuration Utility.<br><br><b>Note:</b> When entering a user name that is a member of an NT domain, separate the domain name and the user name with two backslashes. |

**Step 3** If you changed the comUserAllowed variable, restart the PATROL Agent.

## Configuring DCOM on PATROL Agents

---

**Summary:** In this task you will run the Microsoft DCOM Configuration Utility on each PATROL Agent with which your COM application needs to communicate.

---

### Before You Begin

Perform this task only if you will not configure the PATROL Agent using `wpcnfig` to set the PATROL Agent `comUserAllowed` variable. The `comUserAllowed` variable takes precedence over configuration set by the Microsoft DCOM Configuration Utility. See “Setting PATROL Agent Configuration Variables” on page C-3 for more information.

The PATROL Agent does not support DCOM on versions of Windows prior to Windows NT release 4.0.

Obtain the Microsoft DCOM Configuration Tool (**DCOMCNFG.EXE**) for the operating system and platform on which the agent or agents run. The configuration tool for Windows is distributed with the operating system. Versions for other operating systems are available on the Microsoft World Wide Web site at <http://www.microsoft.com>.

### To Configure DCOM on PATROL Agents

**Step 1** Start the DCOM Configuration tool (refer to the DCOMCNFG documentation for specific instructions on starting the tool).

DCOMCNFG displays a list of applications registered on your computer.

**Step 2** Locate the PATROL Agent on the list of applications. Perform either Step 2.A or 2.B, depending on how your agent is configured.

**2.A** If the agent runs as a service, choose **BMC PATROL Agent Service COM Server** from the list.

**2.B** If the agent runs as a console application, choose **BMC PATROL Agent Console COM Server** from the list.

- Step 3** Refer to the DCOM Configuration Tool documentation for information on additional configuration options offered by the tool.
- Step 4** Click **OK** when you have finished configuring DCOM for the PATROL Agent.
- Step 5** Repeat this procedure for each PATROL agent you to which you need a DCOM interface.

# Configuring Your Computer for DCOM Communication with the PATROL Agent

---

**Summary:** In this task, you will run the PATROL DCOM Registration Utility on the computer that runs your DCOM application.

---

## Before You Begin

You do not need to perform this task if the PATROL Agent is installed on the same computer running your application.

Copy the COM type library file, **IPA.TLB**, to the **\bin** directory under **%PATROL\_HOME%**. The **IPA.TLB** file is supplied with the PATROL 3.3 Agent. Make sure you have administrator privileges before running the PATROL DCOM registration utility. The utility requires administrator-level permissions to modify the computer registry.

## To Configure Your Computer for Communication with the PATROL Agent

**Step 1** Run the PATROL Client Configuration Utility (**PCCFG.EXE**). The executable is located in the **\bin** directory under **%PATROL\_HOME%**.

---

**Note**

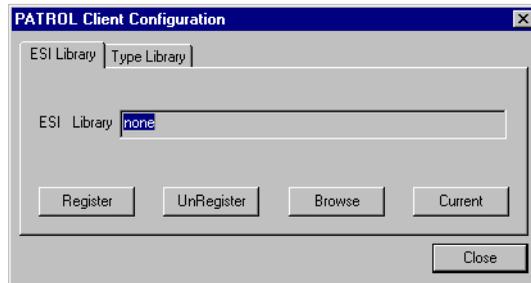
---

Run **PCCFG.EXE** in the same directory where **IPA.TLB** resides

---

The PATROL Client Configuration dialog box displays the currently selected Extended Security Library file.

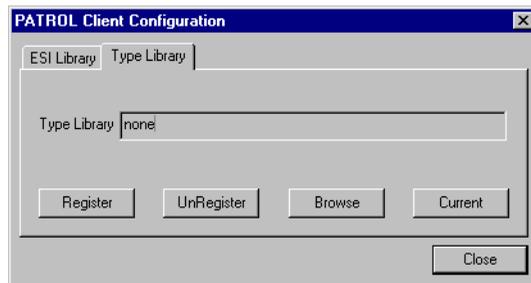
**Figure C-1 PATROL Client Configuration Dialog Box - ESI Library Tab**



**Step 2** Click the **Type Library** tab.

PATROL displays the currently registered COM Type Library file (**IPA.TLB**).

**Figure C-2 PATROL Client Configuration Dialog Box - Type Library Tab**



**Step 3** Click **Browse**, and navigate to directory containing the COM Type Library file.

**Step 4** Select the COM Type Library file, and click **Open**.

**Step 5** Click **Register** to update your computer registry.

**Step 6** Click **Close**.



---

# PATROL SNMP MIB

This appendix describes the Management Information Base (MIB) tables contained in `$PATROL_HOME /lib/patrol.mib`. These tables map PATROL managed objects to SNMP MIB variables. Five tables in **patrol.mib** describe PATROL Agent objects, object variables, applications, application instances, and parameters. This appendix also describes two scalar MIB variables, new syntax types, and the `piV1mTable`.

|  |      |
|--|------|
| MIB Scalar Variables .....             | D-2  |
| <code>objectsCwd</code> .....          | D-2  |
| <code>agentExecuteCommand</code> ..... | D-2  |
| Traps .....                            | D-3  |
| patrol.mib Definitions .....           | D-5  |
| <code>objectsTable</code> .....        | D-6  |
| <code>variablesTable</code> .....      | D-8  |
| <code>applicationsTable</code> .....   | D-11 |
| <code>applInstTable</code> .....       | D-14 |
| <code>parametersTable</code> .....     | D-18 |
| <code>piV1mTable</code> .....          | D-24 |

## MIB Scalar Variables

Two MIB scalar variables are described.

### **objectsCwd**

The **objectsCwd** variable is the parent node for the objectsTable. Only the subnodes of objectsCwd will show in the objectsTable, and only leaves of objects Cwd will show in the variablesTable.

#### **objectsCwd** OBJECT-TYPE

SYNTAX                      DisplayString

ACCESS                      read-write

STATUS                      mandatory

#### DESCRIPTION

“The current node used as a root for the subtree to be examined”

DEFVAL                      {“/”} (start from the global root)

::= { patrolObjects 2 }

### **agentExecuteCommand**

The **agentExecuteCommand** variable implements a kind of RPC where the PATROL Agent is a (PSL) server for an SNMP manager. Setting this variable will execute its value as a PSL script in a specially created PSL process.

#### **agentExecuteCommand** OBJECT-TYPE

SYNTAX                      DisplayString

ACCESS                      read-write

STATUS                      mandatory

#### DESCRIPTION

“Executes the PSL command sent in the same message”

::= { patrolAgent 3 }

# Traps

The PATROL Agent sends traps to registered SNMP managers in V1 format only.

- To register an SNMP manager with the PATROL Agent, add the manager to a row in the piV1mTable.
- To add the manager, send an instance of piV1mRowStatus for the desired ip/port/community with a value of CreateAndGo (4).
- To delete the manager, set its value to Delete (6).

The PATROL Agent sends two types of traps:

- patrolTrapV1StateChanged is sent to all registered SNMP managers when an application instance state changes.
- patrolTrapV1Raised is sent to registered SNMP managers when a PSL snmp\_raise\_std\_trap( ) is called.

The object patrolTrapText carries text passed with the trap. For patrolV1TrapRaised, the value of patrolTrapText.0 is defined by the PSL program as the only parameter for the snmp\_trap\_raise\_std\_trap( ) function. For patrolV1TrapStateChanged, the value of patrolTrapText.0 is a string PATROL\_OBJECT\_NAME NEW\_STATUS or PATROL\_OBJECT\_NAME NEW\_STATUS NEW\_VALUE (PATROL parameters).

Enterprise-specific 1 to 9 are reserved.

## **patrolTrapV1Raised** TRAP-TYPE

ENTERPRISE                    patrolTraps  
OBJECTS                      { patrolTrapText }

### DESCRIPTION

“A patrolTrapV1Raised trap is sent when one of the PSL scripts calls the snmp\_raise\_std\_trap function. This trap is sent to the SNMP V1 entity according to piV1mTable.”

::= 10

### **patrolTrapV1StateChanged** TRAP-TYPE

ENTERPRISE                      patrolTraps  
OBJECTS                          { variableValue }

#### DESCRIPTION

“A patrolTrapV1StateChanged is sent when the PATROL Agent changes the state of the discovered application instances. This trap is sent to the SNMP V1 entity according to piV1mTable.”

::= 11

### **patrolTrapText** OBJECT-TYPE

SYNTAX                          OCTET STRING ( SIZE (0..127))  
ACCESS                          read-only  
STATUS                          mandatory

#### DESCRIPTION

“String passed by PSL to inform the Agent about the trap reason”

::= { patrolTraps 9 }

### **patrolTrapOrigin** OBJECT-TYPE

SYNTAX                          OCTET STRING ( SIZE (0..127))  
ACCESS                          read-only  
STATUS                          optional

#### DESCRIPTION

“Name of the object that originated the trap”

::= { patrolTraps 10 }

### **patrolTrapExtra** OBJECT-TYPE

SYNTAX                          OCTET STRING ( SIZE (0..127))  
ACCESS                          read-only  
STATUS                          optional

#### DESCRIPTION

“Extra information, usually parameter value”

::= { patrolTraps 11 }

## patrol.mib Definitions

bmc                    OBJECT IDENTIFIER ::= { enterprises 1031 }  
patrolMIB            OBJECT IDENTIFIER ::= { bmc 1 }

### Agent Subtree

The Agent subtree is the main subtree for patrol.mib. All manager objects on the Agent side will be in this subtree.

patrolAgent        OBJECT IDENTIFIER ::= { patrolMIB 1 }

### Groups in patrol.mib

patrolObjects     OBJECT IDENTIFIER ::= { patrolAgent 1 }  
patrolTraps        OBJECT IDENTIFIER ::= { patrolAgent 2 }

### New Syntax Types

**PRowStatus**     ::= INTEGER {  
                                  active (1),  
                                  notInService (2),  
                                  notReady (3)  
                                  createAndGo (4)  
                                  createAndWait (5)  
                                  destroy (6)  
                                  }

**OutputMode**     ::= INTEGER {  
                                  none (0),  
                                  text (1),  
                                  gauge (2)  
                                  graph (3)  
                                  console (4)  
                                  }

```

States          ::= INTEGER {
                        ok (0),
                        warn (1),
                        alarm (2)
                        offline (3)
                        void (4)
                    }

```

```

PBoolean       ::= INTEGER {
                        false (0),
                        true (1),
                    }

```

## objectsTable

Each row in the table describes one PATROL Agent object (node in objects variables tree).

### Columns

The table objectsTable contains these columns:

|  |               |
|--|---------------|
| <b>objectName</b>  | DisplayString |
| Name of the object   |               |
| <b>objectDescr</b>   | DisplayString |
| Description of the object, if any                              |               |
| <b>objectRowStatus</b>   | PRowStatus    |
| Row status for the creation of a new row as defined in SNMP V2 |               |

### Objects

The table objectsTable contains the following objects:

**objectsTable** OBJECT-TYPE

|        |                          |
|--------|--------------------------|
| SYNTAX | SEQUENCE OF ObjectsEntry |
| ACCESS | not accessible           |
| STATUS | deprecated               |

DESCRIPTION

“This table is used to get/set PATROL Agent objects. PATROL Agent objects are nodes (and not leaves) of the information tree. Object represents a KM definition.”

::= { patrolObjects 3 }

**objectsEntry** OBJECT-TYPE

|        |                |
|--------|----------------|
| SYNTAX | ObjectsEntry   |
| ACCESS | not accessible |
| STATUS | mandatory      |

DESCRIPTION

“An objectsEntry contains the value and description for a given PSL Agent’s object.”

INDEX { IMPLIED objectName }

::= { objectsTable 1 }

objectsEntry ::= SEQUENCE {  
    objectName  
    objectDescr  
    objectRowStatus

**objectName** OBJECT-TYPE

|        |               |
|--------|---------------|
| SYNTAX | DisplayString |
| ACCESS | read-only     |
| STATUS | mandatory     |

DESCRIPTION

“Name of the mandatory object”

::= { objectsEntry 1 }

**objectDescr** OBJECT-TYPE

SYNTAX                      DisplayString  
ACCESS                        read-write  
STATUS                        mandatory  
DESCRIPTION  
“Textual description of the mandatory object”  
:= { objectsEntry 3 }

**objectRowStatus:** OBJECT-TYPE

SYNTAX                      PRowStatus  
ACCESS                        read-write  
STATUS                        mandatory  
DESCRIPTION  
“Row status for the creation of a new row as defined in SNMP V2”  
:= { objectsEntry 4 }

## variablesTable

Each row describes one PATROL Agent object variable (leaf in objects variables tree).

### Columns

The table variablesTable contains these columns:

**variableName**              DisplayString  
Variable name without the leading path

**variableType**              DisplayString  
A string identifying the type of the variable

**variableValue**             DisplayString  
The variable’s value

**variableDesr**  
Variable help information, if any

**variableRowStatus**        PRowStatus  
Row status for the creation of a new row as defined in SNMP V2

## Objects

The table variablesTable contains these objects:

### **variablesTable** OBJECT-TYPE

SYNTAX SEQUENCE OF VariablesEntry

ACCESS not accessible

STATUS deprecated

#### DESCRIPTION

“This table is used to get/set PATROL Agent object variables. PATROL Agent object variables are leaves (and not nodes) of the information tree.”

::= { patrolObjects 5 }

### **variablesEntry** OBJECT-TYPE

SYNTAX VariablesEntry

ACCESS not accessible

STATUS mandatory

#### DESCRIPTION

“A variablesEntry contains the value and DESCRIPTION (in future) for a given PSL Agent’s object.”

INDEX { IMPLIED variableName }

::= { variablesTable 1 }

variablesEntry ::= SEQUENCE {

variableName

variableType

variableValue

variableDescr

variableRowStatus

### **variableName** OBJECT-TYPE

SYNTAX DisplayString

ACCESS read-only

STATUS mandatory

DESCRIPTION

“A variable name without the path”

::= { variablesEntry 2 }

**variableType** OBJECT-TYPE

SYNTAX DisplayString

ACCESS read-only

STATUS mandatory

DESCRIPTION

“A string describing the variable type. It can be used by a sophisticated SNMP manager to display gauges, graphs, or multimedia measurements.”

::= { variablesEntry 3 }

**variableValue** OBJECT-TYPE

SYNTAX DisplayString

ACCESS read-write

STATUS mandatory

DESCRIPTION

“Current value of the variable”

::= { variablesEntry 4 }

**variableDesr** OBJECT-TYPE

SYNTAX DisplayString

ACCESS read-write

STATUS mandatory

DESCRIPTION

“The variable’s help text”

::= { variablesEntry 5 }

**variableRowStatus** OBJECT-TYPE

|             |  |
|-------------|--|
| SYNTAX      | PRowStatus   |
| ACCESS      | read-write   |
| STATUS      | mandatory  |
| DESCRIPTION | “Row status for the creation of a new row as defined in SNMP V2” |
|             | := { variablesEntry 6 }  |

## applicationsTable

Each row describes one PATROL Agent application.

### Columns

The table applicationsTable contains these columns:

|  |               |
|--|---------------|
| <b>applicationName</b>   | DisplayString |
| The PATROL application name                                    |               |
| <b>applicationState</b>  | States        |
| Application state  |               |
| <b>applWorstInst</b>   | DisplayString |
| Worst instance of the application                              |               |
| <b>applMasterVersion</b>                                       | DisplayString |
| Version number of the application                              |               |
| <b>applMinorRevision</b>                                       | DisplayString |
| Minor revision number of the application                       |               |
| <b>applicationRowStatus</b>                                    | PRowStatus    |
| Row status for the creation of a new row as defined in SNMP V2 |               |
| <b>applicationOid</b>  | INTEGER       |
| (SIZE (0..65535))  |               |

### Objects

The table applicationsTable contains these objects:

**applicationsTable** OBJECT-TYPE

|        |                               |
|--------|-------------------------------|
| SYNTAX | SEQUENCE OF ApplicationsEntry |
| ACCESS | not accessible                |
| STATUS | deprecated                    |

**DESCRIPTION**

“This table is used to get/set PATROL monitored applications. PATROL Agent applications are nodes of the information tree.”

`::= { patrolObjects 6 }`

**applicationsEntry OBJECT-TYPE**

|        |                   |
|--------|-------------------|
| SYNTAX | ApplicationsEntry |
| ACCESS | not accessible    |
| STATUS | mandatory         |

**DESCRIPTION**

“An applicationsEntry contains the value and description for a given PSL Agent’s applications.”

**INDEX** { IMPLIED applicationOid }

`::= { applicationsTable 1 }`

`applicationsEntry ::= SEQUENCE {`  
     applicationName  
     applicationState  
     applicationWorstInst  
     applicationMasterVersion  
     applicationMinorRevision  
     applicationRowStatus  
     applicationOid  
`}`

**applicationName OBJECT-TYPE**

|        |               |
|--------|---------------|
| SYNTAX | DisplayString |
| ACCESS | read-only     |
| STATUS | mandatory     |

**DESCRIPTION**

“Application name”

::= { applicationsEntry 1 }

**applicationState** OBJECT-TYPE

SYNTAX                    States  
ACCESS                    read-only  
STATUS                    mandatory  
DESCRIPTION  
“Application state”

::= { applicationsEntry 2 }

**applWorstInst** OBJECT-TYPE

SYNTAX                    DisplayString  
ACCESS                    read-only  
STATUS                    mandatory  
DESCRIPTION  
“The application’s worst instance”

::= { applicationsEntry 3 }

**applMasterVersion** OBJECT-TYPE

SYNTAX                    INTEGER  
ACCESS                    read-only  
STATUS                    mandatory  
DESCRIPTION  
“The application version number”

::= { applicationsEntry 4 }

**applMinorRevision** OBJECT-TYPE

SYNTAX                    INTEGER  
ACCESS                    read-only  
STATUS                    mandatory  
DESCRIPTION  
“The minor revision number of the application”

::= { applicationsEntry 5 }

### **applicationRowStatus** OBJECT-TYPE

SYNTAX PRowStatus

ACCESS read-only

STATUS mandatory

#### DESCRIPTION

“Row status for the creation of a new row as defined in SNMP V2”

::= { applicationsEntry 6 }

### **applicationOid** OBJECT-TYPE

SYNTAX INTEGER

ACCESS not accessible

STATUS mandatory

#### DESCRIPTION

“Application object id”

::= { applicationsEntry 7 }

## **applInstTable**

Each row describes one PATROL Agent application instance.

### **Columns**

The table applInstTable contains these columns:

**applInstName** DisplayString

PATROL application instance name

**applInstRuleState** States

PATROL application instance state

**applInstStatus** States

PATROL application instance status is the worst of rule state and its worst parameter Status

|   |               |
|---|---------------|
| <b>applInstWorstParam</b>   | DisplayString |
| Application instance's worst parameter  |               |
| <b>applInstCreateIcon</b>   | PBoolean      |
| Create icon for an application instance. Creating an icon with a PATROL Console is at the application level and is equivalent to creating ContainerGadget when the first instance is created. |               |
| <b>applInstRowStatus</b>  | PRowStatus    |
| Row status for the creation of a new row as defined in SNMP V2  |               |
| <b>applInstOid</b>  | INTEGER       |
| (SIZE (0..65535))   |               |
| <b>applInstPInstOid</b>   | OBJECT-TYPE   |
| PATROL application instance's parent object id  |               |
| <b>applInstPAppOid</b>  | OBJECT-TYPE   |
| PATROL application instance's parent application object id  |               |

## Objects

The table `applInstTable` contains these objects:

### **applInstTable** OBJECT-TYPE

SYNTAX SEQUENCE OF `ApplInstEntry`

ACCESS not accessible

STATUS deprecated

#### DESCRIPTION

“This table is used to get/set a PATROL monitored application instance. The PATROL Agent application instances are nodes of the information tree.”

: := { patrolObjects 7 }

### **applInstEntry** OBJECT-TYPE

SYNTAX `ApplInstEntry`

ACCESS not accessible

STATUS mandatory

## DESCRIPTION

“An applInstEntry contains the value and description for a given PSL Agent’s applInst.”

INDEX { applicationOid, applInstOid }

:= { applInstTable 1 }

applInstEntry := SEQUENCE {  
    applInstName  
    applInstRuleState  
    applInstStatus  
    applInstWorstParam  
    applInstCreateIcon  
    applInstRowStatus  
    applInstOid  
    applInstPInstOid  
    applInstPAppOid

### **applInstName** OBJECT-TYPE

SYNTAX DisplayString

ACCESS read-only

STATUS mandatory

## DESCRIPTION

“Application instance name”

:= { applInstEntry 1 }

### **applInstRuleState** OBJECT-TYPE

SYNTAX States

ACCESS read-only

STATUS mandatory

## DESCRIPTION

“Application instance state”

:= { applInstEntry 2 }

### **applInstStatus** OBJECT-TYPE

SYNTAX                               States  
ACCESS                                read-only  
STATUS                                 mandatory

#### DESCRIPTION

“The application instance status is the worst of rule state and its worst parameter’s status.”

::= { applInstEntry 3 }

### **applInstWorstParam** OBJECT-TYPE

SYNTAX                               DisplayString  
ACCESS                                read-only  
STATUS                                 mandatory

#### DESCRIPTION

“Application instance’s worst parameter”

::= { applInstEntry 4 }

### **applInstCreateIcon** OBJECT-TYPE

SYNTAX                               PBoolean  
ACCESS                                read-only  
STATUS                                 mandatory

#### DESCRIPTION

“True means to create icons for application instances. False means that it is not necessary to create icons for the instances.”

::= { applInstEntry 5 }

### **applInstRowStatus** OBJECT-TYPE

SYNTAX                               PRowStatus  
ACCESS                                read-only  
STATUS                                 mandatory

#### DESCRIPTION

“Row status for the creation of a new row as defined in SNMP V2”

::= { applInstEntry 6 }

**applInstOid** OBJECT-TYPE

SYNTAX                    INTEGER  
ACCESS                    not accessible  
STATUS                    mandatory  
DESCRIPTION  
“Application instance object id”

::= { applInstEntry 7 }

**applInstPOid** OBJECT-TYPE

SYNTAX                    INTEGER  
ACCESS                    not accessible  
STATUS                    mandatory  
DESCRIPTION  
“Application instance’s parent object id”

::= { applInstEntry 8 }

**applInstPAppOid** OBJECT-TYPE

SYNTAX                    INTEGER  
ACCESS                    not accessible  
STATUS                    mandatory  
DESCRIPTION  
“Application instance’s parent application id”

::= { applInstEntry 9 }

## parametersTable

Each row describes one PATROL Agent parameter.

## Columns

The table parametersTable contains these columns:

|   |               |
|---|---------------|
| <b>parameterName</b>  | DisplayString |
| The parameter name (DisplayString)  |               |
| <b>parameterState</b>   | States        |
| Parameter's state   |               |
| <b>parameterCurrentTime</b>   | TimeTicks     |
| Time stamp when the current parameter value was produced  |               |
| <b>parameterCurrentValue</b>  | DisplayString |
| Parameter's current value   |               |
| <b>parameterPollingInt</b>  | INTEGER       |
| Parameter's polling interval  |               |
| <b>parameterRetries</b>   | INTEGER       |
| Parameter's timeout retries   |               |
| <b>parameterOutputMode</b>  | OutputMode    |
| Parameter's output mode   |               |
| <b>parameterAutoScale</b>   | PBoolean      |
| True(1) means to use automatic scale. False(0) means to use the following YaxisMin and YaxisMax fields to draw scale. |               |
| <b>parameterYaxisMin</b>  | INTEGER       |
| Minimum value for drawing the y-axis scale if the parameter AutoScale is false (0)                                    |               |
| <b>parameterYaxisMax</b>  | INTEGER       |
| Maximum value for drawing the y-axis scale if the parameterAutoScale is false(0)                                      |               |
| <b>parameterRowStatus</b>   | PRowStatus    |
| Row status for creating a new row as defined in SNMP V2   |               |
| <b>parameterObjId</b>   | INTEGER       |
| (SIZE (0..65535))   |               |
| <b>parameterActiveStat</b>  | PBoolean      |
| Boolean value of the parameter explaining whether the parameter was created as active                                 |               |
| <b>parameterRunningStat</b>   | PBoolean      |
| Boolean value indicating the parameter is either running or suspended for the current instance                        |               |

## Objects

The table parametersTable contains these objects:

### **parametersTable** OBJECT-TYPE

|        |                             |
|--------|-----------------------------|
| SYNTAX | SEQUENCE OF ParametersEntry |
| ACCESS | not accessible              |
| STATUS | deprecated                  |

#### DESCRIPTION

“This table is used to get/set PATROL application instance monitored parameters. PATROL Agent parameters are leaves of the information tree.”

: := { patrolObjects 8 }

### **parametersEntry** OBJECT-TYPE

|        |                 |
|--------|-----------------|
| SYNTAX | ParametersEntry |
| ACCESS | not accessible  |
| STATUS | mandatory       |

#### DESCRIPTION

“The value and description for a given PSL Agent’s parameters”

INDEX { applicationOid, applInstOid, parameterObjId }

: := { parametersTable 1 }

parametersEntry : := SEQUENCE {

- parameterName
- parameterState
- parameterCurrentTime
- parameterCurrentValue
- parameterPollingInt
- parameterRetries
- parameterOutputMode
- parameterAutoScale
- parameterYaxisMin
- parameterYaxisMax

parameterRowStatus  
parameterObjId  
parameterActiveStat  
parameterRunningStat

**parameterName** OBJECT-TYPE

SYNTAX DisplayString  
ACCESS read-only  
STATUS mandatory  
DESCRIPTION  
"Parameter name"  
 ::= { parametersEntry 1 }

**parameterState** OBJECT-TYPE

SYNTAX States  
ACCESS read-only  
STATUS mandatory  
DESCRIPTION  
"Parameter state"  
 ::= { parametersEntry 2 }

**parameterCurrentTime** OBJECT-TYPE

SYNTAX TimeTicks  
ACCESS read-only  
STATUS mandatory  
DESCRIPTION  
"Time stamp when the current value was produced"  
 ::= { parametersEntry 3 }

**parameterCurrentValue** OBJECT-TYPE

SYNTAX DisplayString  
ACCESS read-only

STATUS mandatory  
DESCRIPTION  
“Parameter’s current value”  
:= { parametersEntry 4 }

**parameterPollingInt** OBJECT-TYPE

SYNTAX INTEGER  
ACCESS read-only  
STATUS mandatory  
DESCRIPTION  
“Parameter polling interval”  
:= { parametersEntry 5 }

**parameterRetries** OBJECT-TYPE

SYNTAX INTEGER  
ACCESS read-only  
STATUS mandatory  
DESCRIPTION  
“Parameter timeout retries”  
:= { parametersEntry 6 }

**parameterOutputMode** OBJECT-TYPE

SYNTAX OutputMode  
ACCESS read-only  
STATUS mandatory  
DESCRIPTION  
“Parameter output mode”  
:= { parametersEntry 7 }

**parameterAutoScale** OBJECT-TYPE

SYNTAX PBoolean  
ACCESS read-only

STATUS mandatory

DESCRIPTION

“True(1) means to use automatic scale. False(0) means to use the following YaxisMin and YaxisMax fields to draw y-axis scale.”

:= { parametersEntry 8 }

**parameterYaxisMin** OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-only

STATUS mandatory

DESCRIPTION

“Minimum value for drawing y-axis scale if parameterAutoScale is false(0)”

:= { parametersEntry 9 }

**parameterYaxisMax** OBJECT-TYPE

SYNTAX INTEGER

ACCESS read-only

STATUS mandatory

DESCRIPTION

“Maximum value for drawing y-axis scale if parameterAutoScale is false(0)”

:= { parametersEntry 10 }

**parameterRowStatus** OBJECT-TYPE

SYNTAX PRowStatus

ACCESS read-write

STATUS mandatory

DESCRIPTION

“Row status for creating a new row as defined in SNMP V2”

:= { parametersEntry 11 }

**parameterObjId** OBJECT-TYPE



|   |              |
|---|--------------|
| <b>piV1mIp</b><br>IP address            | IpAddress    |
| <b>piV1mPort</b><br>(SIZE (0..65535))   | INTEGER      |
| <b>piV1mCommunity</b><br>SIZE (0..127)) | OCTET STRING |
| <b>piV1mRowStatus</b>                   | PRowStatus   |

Row status for the creation of a new row as defined in SNMP V2

## Objects

The table piV1mTable contains these objects:

### piV1mTable OBJECT-TYPE

|        |                        |
|--------|------------------------|
| SYNTAX | SEQUENCE OF PiV1mEntry |
| ACCESS | not accessible         |
| STATUS | mandatory              |

#### DESCRIPTION

“This table keeps information about SNMP V1 managers interested in receiving traps from the PATROL Agent.”

::= { patrolTraps 10 }

### piV1mEntry OBJECT-TYPE

|        |                |
|--------|----------------|
| SYNTAX | PiV1mEntry     |
| ACCESS | not accessible |
| STATUS | mandatory      |

#### DESCRIPTION

“A piV1mTable describes one SNMP V1 manager interested in getting traps from the PATROL Agent.”

INDEX { piV1mIp, piV1mPort, piV1mCommunity }

::= { piV1mTable 1 }

piV1mEntry ::= SEQUENCE {

piV1mIp

piV1mPort

piV1mCommunity

piV1mRowStatus

**piV1mIp** OBJECT-TYPE

SYNTAX                    IpAddress  
ACCESS                    not accessible  
STATUS                    mandatory  
DESCRIPTION  
“IP address of the interested manager”  
:= { piV1mEntry 1 }

**piV1mPort** OBJECT-TYPE

SYNTAX                    INTEGER  
ACCESS                    not accessible  
STATUS                    mandatory  
DESCRIPTION  
“Port number of the interested SNMP manager”  
DEFVAL                    { 161 }  
:= { piV1mEntry 2 }

**piV1mCommunity** OBJECT-TYPE

SYNTAX                    OCTET STRING  
ACCESS                    not accessible  
STATUS                    mandatory  
DESCRIPTION  
“Community string for the interested SNMP manager”  
DEFVAL                    { public }  
:= { piV1mEntry 3 }

**piV1mRowStatus** OBJECT-TYPE

SYNTAX                    PRowStatus  
ACCESS                    read-write

STATUS

mandatory

DESCRIPTION

“Row status for the creation of a new row as defined in SNMP V2”

:= { piV1mEntry 4 }



---

# Power Management Support

This appendix briefly describes the PATROL Agent's support of application power management. It defines OnNow and ACPI and describes how they work. It then describes how the PATROL Agent supports OnNow/ACPI. This appendix contains the following sections:

|   |     |
|---|-----|
| OnNow/ACPI Support .....                  | E-2 |
| ACPI Definition and Functions .....       | E-2 |
| OnNow Definition and Functions .....      | E-2 |
| Sleep State .....                         | E-3 |
| OnNow/ACPI Operation .....                | E-3 |
| PATROL Agent Support for OnNow/ACPI ..... | E-4 |

# OnNow/ACPI Support

The PATROL Agent supports OnNow/ACPI power management on Windows platforms.

## ACPI Definition and Functions

ACPI is the Advanced Configuration and Power Interface. It succeeds Application Power Management (APM) and extends its capabilities. ACPI facilitates power management by making hardware status information available to the operating system. ACPI enables computers to

- manage power consumption, especially on portables, by turning on its peripherals and off
- be turned on and off by external devices or placed into a sleep state, so that the touch of a mouse or the press of a key will “wake up” the machine

## OnNow Definition and Functions

OnNow is Microsoft’s implementation of the Advanced Configuration and Power Interface (ACPI). It takes advantage of PC motherboards with BIOSes that support ACPI.

The OnNow feature allows a computer to always be on, and therefore instantly accessible to users, while consuming very little power when not in use.

## Sleep State

Sleep state is a condition of power management that enables a system to suspend all processes. This suspension is a virtual shut down that permits a system to awaken and respond to commands without consuming much power. Sleep states include

- hibernate
- stand by
- sleep

## OnNow/ACPI Operation

The following table describes how OnNow/ACPI-compliant applications and non-compliant applications behave when a system enters into sleep state and then awakens.

| Order | Description  |
|-------|--|
| 1     | An internal or external device issues a command to go into sleep state.  |
| 2     | The OnNow/ACPI function takes a snap shot of the system's state and all its applications by writing everything in memory to the hard drive.  |
| 3     | Applications that support power options shut down their processes or die gracefully. Applications that do not support power options simply fail.   |
| 4     | An internal or external device issues an awaken command.   |
| 5     | The OnNow/ACPI function reads the information written to the disk and restores the state of OnNow-supported applications. It also attempts to restore the state of applications that do not support power management. These applications may not be recoverable because their processes are looping or suspended or startup information is unavailable because its owning data structures (such as files or databases) were corrupted during the shutdown. |

## PATROL Agent Support for OnNow/ACPI

The OnNow/ACPI capabilities built into the PATROL Agent include:

- registering with the system during start up to receive power events; once enabled, the PatrolAgent service responds and processes the SERVICE\_CONTROL\_POWEREVENT control requests
- properly shutting down all processes when the system is placed in a sleeping state; this procedure reduces the chances that applications and files will be corrupted
- restarting the PATROL Console and PATROL Agent when the system is awakened and re-establishing connections without user intervention

---

# Support for Regular Expressions

This appendix describes what metacharacters the PATROL Agent uses for regular expressions and where regular expressions are supported by the agent. This appendix contains the following sections:

|  |     |
|--|-----|
| Regular Expressions . . . . .                        | F-2 |
| Components that Support Regular Expressions. . . . . | F-2 |
| Benefits . . . . .                                   | F-3 |
| Filename Metacharacters and Wildcards . . . . .      | F-3 |
| Using Character Classes [ ] . . . . .                | F-5 |
| Using Quantifiers (*, ?, +) . . . . .                | F-6 |
| Using Anchors (\$, ^, \<, \>) . . . . .              | F-7 |

# Regular Expressions

The PATROL Agent supports the limited use of filename metacharacters (Unix) or wildcards (Windows) while creating regular expressions. Regular expressions enable you to take advantage of naming conventions and other predictable output. With these expressions, you can perform pattern matching based on text strings or numbers.

## Components that Support Regular Expressions

PATROL supports regular expressions for the following components:

- PSL functions
  - `grep()`
  - `dump_hist()`
- Inclusion/exclusion lists for instances
- `dump_hist` utility (command line arguments)
- Agent Configuration Variables
  - `accessControlList`
  - `disabledKMs`
  - `disabledKMsArch`

---

### Note

---

PATROL supports a subset of regular expressions defined by the REGEXP standard (REGEX API). Do not confuse PATROL supported subsets with the regular expressions used by Unix commands such as `grep`.

---

## Benefits

The benefits of using regular expressions include

- a reduction in the number of entries required for an exhaustive list of items with uniform naming conventions
- a means to circumvent the 1024 character size limitation for Agent configuration variables

## Filename Metacharacters and Wildcards

Filename metacharacters, or wildcards as they are referred to in Windows, describe all the variations of a given pattern. These characters do not match themselves. Instead, they describe the repetition and occurrence of other characters or groups of characters.

**Table F-1 Wildcards Supported by PATROL Regular Expressions**

| Symbol                   | Description   | Example                      | Possible Results   |
|--------------------------|---|------------------------------|--|
| <b>Atom</b>              |   |                              |  |
| .                        | match any single character  | user.                        | user1, users   |
| <b>Quantifiers</b>       |   |                              |  |
| *                        | match the preceding element (group of characters or number) zero or more times  | b*                           | " " (empty string), b, bbbb                                    |
|                          |   | host.*                       | host, hostname, host_45  |
| ?                        | match the preceding element zero or one time  | 5?                           | " " (empty string), 5  |
|                          |   | user.?                       | user, users, user5   |
| +                        | match the preceding element one or more times   | z+                           | z, zzz, zzzzzz   |
|                          |   | admin.+                      | administrator, admin1  |
| <b>Character Classes</b> |   |                              |  |
| [ ]                      | match enclosed characters. Commas (,) separate individual characters. Hyphens (-) indicate alphabetic and numeric ranges. | [a,b][2-5]                   | a2,a3, b4  |
|                          |   | .*[3,5,7]                    | host_3, host_5, admin7   |
| <b>Anchor</b>            |   |                              |  |
| \$                       | match at end of line  | terminate.\$                 | (all lines that end with "terminate.")                         |
| ^                        | match at start of line  | ^[eE]rror                    | (all lines that begin with "Error" or "error")                 |
| \<                       | match beginning of word   | \<l                          | load, level, limit   |
| \>                       | match end of word   | \>x                          | fax, hex, box  |
| <b>Escape and Group</b>  |   |                              |  |
| \                        | escape a wildcard character so that the expression uses its literal value in comparisons                                  | license\..*                  | license.nt_server, license.bsmith, licence.                    |
| ()                       | group regular expressions and determine the order of precedence   | (ERRMSG.?)^ w.?<br>generated | ERRMSG 44789<br>ERRMSG 78943<br>ERRMSG 89991<br>were generated |

## Using Character Classes [ ]

The square brackets construct specifies a set of characters (a character class) of which one matches. Character classes consist of text characters, integers, and whitespace characters.

### Examples

The following examples display a number of ways to employ character classes to create a variety of patterns.

| <b>Pattern</b> | <b>Description</b>   |
|----------------|--|
| 1[0-9]         | matches any 2-digit number beginning with a such as 11, 15, 18   |
| [1-3][0-9]     | matches any 2-digit number beginning with 1, 2 or 3 such as 10, 39, 24   |
| [2,4,6][a,b,z] | matches 2-digit alphanumeric character beginning with 2, 4, or 6 and ending in a, b, or z such as 2z, 4a, 6b, 6z |
| [a-zA-Z]       | matches any upper or lower case letter such as a Y k L   |
| [eE] mail      | matches capitalized and lower case occurrences of the word "email"   |

## Using Quantifiers (\*, ?, +)

The quantifiers specify how many instances of the previous element can match. The expression `.`\* instructs the system to match any single character (represented by the period `.`) any number of times (represented by the asterisk `*`). Using patterns before and after quantifiers makes the search more precise. This concept is similar to searching for words based on prefixes and suffixes in English.

### Examples

The following examples display a number of ways to employ quantifiers to create a variety of patterns.

| Pattern                 | Description   |
|-------------------------|---|
| <code>e*</code>         | matches any string of zero or more “e”s such as an empty string, e, eee   |
| <code>1+</code>         | matches any string of one or more “1”s such as 1, 111, 1111111  |
| <code>admin.*</code>    | matches any string that begins with “admin” and contains zero or more characters such as administrator, admin_1, admin-bsmith, admin.                   |
| <code>NT.+</code>       | matches any string that begins with “NT” and contains at least one or more characters such as administrator, NT_CPU, NTkm_version, NT_, but not NT.     |
| <code>admin.?</code>    | matches any string that begins with “admin” and contains zero or one more character such as admin, admin1, admin2, adminZ                               |
| <code>e.*Houston</code> | matches any string that begins with an e and ends with Houston such as email_administrator-Houston, ebiz-Houston, ecommerce security advisor in Houston |
| <code>.?Smith</code>    | matches any string that begins with zero or one occurrences of any character and ends with “Smith” such as BSmith, 1Smith, _Smith, Smith                |

## Using Anchors (\$, ^, \<, \>)

The anchors specify a position within a string. They are used in conjunction with quantifiers and character classes to find patterns in a specific location.

### Examples

The following examples display a number of ways to employ anchors to create a variety of patterns. The anchor examples use character classes and quantifiers to make them more meaningful.

| Pattern                   | Description   |
|---------------------------|---|
| <code>^[abcABC]</code>    | matches any line that begins with “a”, “b”, or “c” such as a<br>Abort?; Bad disk please insert a valid disk; clusters<br>verified = 231 |
| <code>\$\.</code>         | matches any line that ends with a period (.)  |
| <code>\&lt;[sS]</code>    | matches any word that begins with an upper or lower case<br>letter “s” such as system, Security, storage                                |
| <code>_server\&gt;</code> | matches any word that ends in “_server” such as<br>oracle8_server, sql-test_server, nt_server   |



---

# Managing Unix Environment Variables

This appendix describes how you can store Unix environment variables in a shell script file that will not be overwritten by subsequent installations of the PATROL Agent. This appendix contains the following sections:

|   |     |
|---|-----|
| Storing Environment Variables in a Script File .....  | G-2 |
| Purpose .....   | G-2 |
| Operation .....                                       | G-2 |
| Filename and Format .....                             | G-3 |
| Example .....   | G-4 |
| Set Environment Before Running PATROL Utilities ..... | G-4 |
| Exception .....                                       | G-5 |

# Storing Environment Variables in a Script File

The PATROL Agent allows you to store all your environment variables in a user-defined shell script file. This file is read and sourced as the last step of the startup process. It can contain any valid shell script code which includes, but is not limited to, initialization and modification of environment variables.

## Purpose

The support for this script file provides a means to customize your PATROL environment without modifying the scripts provided by PATROL.

In previous versions of PATROL, installing PATROL on top of an existing implementation replaced the scripts (**Patrol**, **PatrolAgent**, **emcons**, **patrolrc.sh**, and others). Any customizations made to these script files were lost. The installation of PATROL 3.4 also replaces all of these scripts, but it does not overwrite your customized file.

## Operation

The PATROL Agent inherits the environment of the shell in which it is started. The PATROL startup scripts, **./patrolrc.sh** (Korn and Bourne shell) and **./Patrolrc** (C shell), treat your user-defined shell script file as optional. If the user-defined shell script file exists, the startup script sources it after all other PATROL script processing. If it does not exist, the script continues. The startup script (**patrolrc.sh** and **Patrolrc**, respectively) is read by the **./Patrol** and **./PatrolAgent** scripts, which are usually used to start the console and agent, respectively.

---

### Warning

---

To guarantee that the PATROL Agent and PATROL Console source the user-defined scripts (**userrrc.sh** and **.userrrc**) during startup, source the appropriate startup scripts (**./patrol.sh** and **./Patrolrc**) before starting the agent and console.

---

## Filename and Format

PATROL supports custom shell scripts on both Korn\Bourne shell and C shell.

---

### Note

---

You must create the appropriate script file for your environment. The installation process does not create it.

---

## Bourne Shell and Korn Shell

### Filename

```
userrc.sh
```

### Variable Format

```
variableName1=value1  
variableName2=value2  
export variableName1  
export variableName2
```

## C Shell

### Filename

```
.userrc
```

### Variable Format

```
setenv variableName1=value1  
setenv variableName2=value2
```

## Example

The following Bourne\Korn script defines and exports several variables:

```
HOME=X:
PATROL_HOME=$HOME/patrol
PATROL_CFG=$PATROL_HOME/config
PATROL_CACHE=$HOME/patrol

export HOME
export PATROL_HOME
export PATROL_CFG
export PATROL_CACHE
```

## Set Environment Before Running PATROL Utilities

Prior to building PEMAPI binaries with “make,” or running PATROL utility such as pconfig and xpconfig, PatrolCli, and PatrolLink, run the following scripts:

### Bourne Shell and Korn Shell

```
./patrolrc.sh
```

### C Shell

```
source .patrolrc
```

## Exception

Under certain circumstances, the PATROL 3.5 installation program will create the script file **userrc.sh** or append to it if it already exists. The following conditions must be met:

- The PATROL Agent is being installed.
- Agent Port number 3181 is specified during the installation.
- The PATROL\_PORT environment variable is NOT defined.

---

### Note

---

This exception applies only to Bourne\Korn shell.

---

## Contents

If the three conditions are met, the installation program either creates a new **userrc.sh** file and writes the following lines in it or appends them to an existing file.

```
PATROL_PORT=3181
export PATROL_PORT
```



---

# Managing Performance

This appendix describes how to manage the performance of the PATROL Agent. It describes how to alter some of agent configuration variables to affect the PATROL Agent jobs and achieve better performance. It also describes how to define the number of PSL instructions the agent performs for a process within a certain period of time, how often the agent attempts to discover applications, how to set the execution priority of processes, and how to balance the agent's load.

---

## Warning

---

The variables discussed in this appendix can significantly affect the performance of the PATROL Agent. Before changing any tuning variables in a production environment, contact your PATROL customer support representative.

---

This appendix discusses the following topics:

|  |     |
|--|-----|
| PATROL Agent Mode of Operation . . . . .   | H-3 |
| Main Run Queue . . . . .                   | H-3 |
| PSL Run Queue . . . . .                    | H-3 |
| Recurring or Periodic Jobs . . . . .       | H-4 |
| Scheduling Interval . . . . .              | H-4 |
| Setting PSL Instruction Limits . . . . .   | H-4 |
| PSL Instruction Limits Operation . . . . . | H-4 |
| Exceeding the Limit. . . . .               | H-5 |
| Internal Delay. . . . .                    | H-5 |
| Increasing the Limit. . . . .              | H-5 |
| Maximum Number of Instructions. . . . .    | H-6 |

|   |      |
|---|------|
| Time Period for Instruction Execution . . . . .             | H-7  |
| Setting Refresh Rate and Discovery Cycle . . . . .          | H-8  |
| Relationship Between Cache and Discovery . . . . .          | H-8  |
| Process Cache Refresh Rate . . . . .                        | H-9  |
| Application Discovery Rate . . . . .                        | H-9  |
| Setting Operating System Scheduling Priority . . . . .      | H-10 |
| Priority Range . . . . .                                    | H-10 |
| For PATROL Agent . . . . .                                  | H-12 |
| For Processes Created by PATROL Agent . . . . .             | H-12 |
| For Cache Refresh Process . . . . .                         | H-13 |
| For Internal Cache Refresh Process . . . . .                | H-13 |
| Load Balancing the PATROL Agent . . . . .                   | H-15 |
| PATROL Agent's Scheduling Policy . . . . .                  | H-15 |
| Interval Between Internal Processes . . . . .               | H-17 |
| Increment Checking of Internal Process Interval . . . . .   | H-17 |
| Maximum Delay of an Internal Process . . . . .              | H-18 |
| Controlling the Number of Processes . . . . .               | H-19 |
| Limit the Number of Files Opened Simultaneously . . . . .   | H-19 |
| Limit the Number of Processes Run Simultaneously . . . . .  | H-20 |
| Limiting the Number of patrolperf Instances . . . . .       | H-20 |
| Selecting PSL Debugging Level . . . . .                     | H-21 |
| Set PSL Runtime Error Level . . . . .                       | H-21 |
| Managing Performance Retrieval Parameters . . . . .         | H-22 |
| Establishing a Timeout Period . . . . .                     | H-22 |
| Determining the Maximum Number of Data Points Collected     |      |
| Before Restarting . . . . .                                 | H-23 |
| Specifying the Debug File . . . . .                         | H-23 |
| Overriding Settings During a Session . . . . .              | H-24 |
| Resetting Values . . . . .                                  | H-24 |
| Managing the PATROL Agent from the PATROL Console . . . . . | H-24 |

# PATROL Agent Mode of Operation

The PATROL Agent monitors applications by executing various jobs on a periodic basis. These jobs include

- running various parameters scripts
- refreshing process cache
- performing application discovery

The agent has two internal run queues which process all jobs: Main and PSL. If a job is executing, it cannot be in a queue.

## Main Run Queue

The main run queue manages all jobs. When it is time for a job to be scheduled for execution, the agent uses an algorithm to calculate its time of execution and places it in the queue. The algorithm takes into consideration the run time of other jobs and various agent scheduling variables. At execution time, the agent does one of the following:

- spawns a child process to execute as a separate OS process, if it is an operating system command
- places the job in the PSL run queue, if it is a PSL command

## PSL Run Queue

The PSL run queue manages all PSL commands. It runs the commands in the agent's process space. The PSL run queue works by alternating between commands in a round-robin fashion. During each processing segment, the agent performs as many PSL instructions as it can, up to 50, within the allotted time period.

## Recurring or Periodic Jobs

Once a periodic job terminates, the agent reschedules the job and places it back in the main run queue.

## Scheduling Interval

PATROL Agent job scheduling is inexact. The agent schedules jobs to run as close as possible to their “ideal” execution time without overburdening the system with PATROL activity.

In determining how often a job runs, you need to figure out how long you can wait to gather statistics. The longer PATROL monitoring activities can be postponed, the less the impact on the system.

## Setting PSL Instruction Limits

The PATROL Agent is a single-threaded program that executes a set number of instructions for a process. If the process does not finish after those instructions have been completed, the agent suspends that process and executes one or more other processes. Then, based on the agent’s internal schedule, it returns to finish the suspended process.

The PSL instruction limits work within this mode of operation to ensure that instruction-intensive processes do not monopolize the agent.

## PSL Instruction Limits Operation

The PATROL Agent instruction limits control how many PSL instructions the agent will execute within a certain time period for a particular PSL process before it “slows down” the speed with which it executes instructions for that process. The agent slows down its execution of a particular process by inserting a delay into the internal schedule. The delay increases the amount of time that the agent spends on other processes, reducing the chances that the agent will execute the maximum number of instructions within the specified time period.

## Exceeding the Limit

If a process exceeds the PSL instruction limits, the agent

- writes to the PATROL error log the message, “PSL may be in an infinite loop”
- slows down the speed with which it executes the process as described in “PSL Instruction Limits Operation”

## Internal Delay

The agent slows down the processing of a PSL command by adding an internal scheduling delay of 200 milliseconds. The agent adds this delay to the amount of time between instruction execution cycles as it goes round-robin style from PSL command to PSL command in the PSL run queue.

Each time the process exceeds the limit during the same execution cycle, the agent adds another delay of 200 milliseconds. This behavior continues until the execution cycle ends. After the delay, the job is returned to the PSL run queue and uses the standard interval.

## Increasing the Limit

The PSL instruction limit is a ratio of instructions per time period. To increase the limit and reduce the chances of incurring a delay, you can either

- increase the amount of instructions
- decrease the time period

---

### Note

---

When monitoring applications with many instances, you may need to increase the default settings.

---

## Maximum Number of Instructions

The `/AgentSetup/AgentTuning/pslInstructionMax` configuration variable specifies the maximum number of instructions that the agent can execute for a single PSL process within a certain time period without incurring an internal scheduling delay.

To turn off the delay, set `pslInstructionMax` to 0 (zero).

|                                |  |
|--------------------------------|--|
| <b>Format and Type of Data</b> | numeric, instructions  |
| <b>Default Value</b>           | 500000   |
| <b>Minimum and Maximum</b>     | 0, none  |
| <b>Dependencies</b>            | <code>/AgentSetup/AgentTuning/pslInstructionPeriod</code> specifies the time period within which the instructions must be executed |
| <b>Recommendation</b>          | none   |

## Time Period for Instruction Execution

The `/AgentSetup/AgentTuning/pslInstructionPeriod` configuration variable specifies the period of time in seconds that the agent can execute the maximum number of instructions for a single PSL process without incurring an internal scheduling delay.

A value of 0 for `pslInstructionPeriod` means that the PSL process can execute without incurring a delay.

|                                |  |
|--------------------------------|--|
| <b>Format and Type of Data</b> | numeric, seconds   |
| <b>Default Value</b>           | 7200 (12 minutes)  |
| <b>Minimum and Maximum</b>     | 0, none  |
| <b>Dependencies</b>            | <code>/AgentSetup/AgentTuning/pslInstructionMax</code> specifies the maximum number of PSL instructions the agent can execute in the time period designated by this variable |
| <b>Recommendation</b>          | none   |

# Setting Refresh Rate and Discovery Cycle

To monitor an application, the PATROL Agent must first discover the application and learn its current status. Discovery takes time and consumes system resources. To lessen the affect of discovery on system performance during discovery, the agent takes a snapshot of the system's process table and stores it in an internal memory structure referred to as the process cache. When the agent runs a discovery script for an application, the script reads the information from the process cache. Because the cache is in memory, the process takes less time and uses less resources than querying the system. The cache is periodically updated at a user-defined interval. This update is referred to as the refresh.

The first time an application discovery script runs after the process cache has been refreshed is called full discovery. Each subsequent discovery after a full discovery is called a partial discovery. The agent continues to run partial discoveries until the cache is refreshed.

During the partial application discovery cycle, the discovery script queries the system to check only those processes that are vital to that application.

---

## Note

---

It is possible to have both pre-discovery and discovery PSL scripts running at the same time.

---

## Relationship Between Cache and Discovery

The `getProcsCycles` variable governs the rate at which the PATROL Agent Discovery Cache is refreshed. Each time the cache is refreshed, the agent runs a full application discovery. Any new applications and state changes are then displayed. The `applCheckCycle` variable determines how often the agent runs a partial application discovery between refreshes.

## Process Cache Refresh Rate

The `/AgentSetup/AgentTuning/getProcsCycle` configuration variable specifies the process cache refresh rate in seconds.

|                                |   |
|--------------------------------|---|
| <b>Format and Type of Data</b> | numeric, seconds  |
| <b>Default Value</b>           | 300   |
| <b>Minimum and Maximum</b>     | 20, none  |
| <b>Dependencies</b>            | <code>/AgentSetup/AgentTuning/procCacheSchedPriority</code><br><code>/AgentSetup/AgentTuning/procCachePriority</code><br><br>If either of these variables has a low priority, the cache refresh may be slightly delayed because other processes take priority in the OS or the agent. For example, the cache may be refreshed at intervals of 310 seconds, 305 seconds, and 300 seconds depending upon pending and running processes. |
| <b>Recommendation</b>          | none  |

## Application Discovery Rate

The `/AgentSetup/AgentTuning/applCheckCycle` configuration variable specifies the interval at which the agent runs partial application discoveries.

|                                |                  |
|--------------------------------|------------------|
| <b>Format and Type of Data</b> | numeric, seconds |
| <b>Default Value</b>           | 40               |
| <b>Minimum and Maximum</b>     | 10, none         |
| <b>Dependencies</b>            | none             |
| <b>Recommendation</b>          | none             |

# Setting Operating System Scheduling Priority

The scheduling priority is equivalent to the nice command in Unix. It sets the execution priority for a process, which determines in what order the operating system executes processes.

## Priority Range

In general, the priority range spans from 0 to 20. However, different operating systems interpret these values differently.

### Unix, OS/2

The lower the value, the higher the priority (faster execution). To raise the priority of the agent to high, you must enter a negative number. However, raising the priority to high is not advisable because it uses significantly more system resources and can cause other applications to crash.

|                  |                 |
|------------------|-----------------|
| <b>11 to 20</b>  | low priority    |
| <b>0 to 10</b>   | normal priority |
| <b>-1 to -15</b> | high priority   |

---

### Example

---

In the operating system processing queue, a process with a scheduling priority of 2 runs before a process of 8, which runs before a process of 13.

---

These systems have a default process priority. The PATROL priority is added to the default priority to determine the agent's process priority on the OS.

default process priority + agentPriority = agent process priority on the OS

---

### Example

---

A Unix box's default process priority is 10. The PATROL Agent installed on the box has an agent priority of 5. Therefore, the priority of the agent process on the OS is 15.

---

## Windows

This configuration variable does not affect the priority of the agent process on the Windows platform. The priority is directly proportional to the value; the higher the number, the higher the priority. The agent's default priority is normal. To change the agent process's priority, you must use the Windows Task Manager.

Windows priorities in order of highest to lowest are

- Realtime
- High
- Normal
- Low

## OpenVMS

The higher the value, the higher the priority (faster execution). The current process's priority is the default priority.

---

### Example

---

In the OpenVMS operating system processing queue, a process with a scheduling priority of 14 runs before a process of 8, which runs before a process of 3.

---

## On Alpha

Priorities range from 0 to 63, where 63 is the highest.

**0-15** are normal priorities

**16-63** are real-time priorities

### On VAX

Priorities range from 0 to 31, where 31 is the highest.

**0-15** are normal priorities

**16-31** are real-time priorities

## For PATROL Agent

The `/AgentSetup/AgentTuning/agentPriority` configuration variable specifies the operating system scheduling priority for the PATROL Agent process.

|                                |   |
|--------------------------------|---|
| <b>Format and Type of Data</b> | numeric, not applicable   |
| <b>Default Value</b>           | 10  |
| <b>Minimum and Maximum</b>     | 0, 20   |
| <b>Override</b>                | none  |
| <b>Dependencies</b>            | none  |
| <b>Recommendation</b>          | The agent process is given a medium priority so that it does not interfere with the processes of the application that it is monitoring. |

## For Processes Created by PATROL Agent

The `/AgentSetup/AgentTuning/userPriority` configuration variable specifies the operating system scheduling priority for processes created by the PATROL Agent. These processes are external processes that the agent spawns from the main run queue.

|                                |                         |
|--------------------------------|-------------------------|
| <b>Format and Type of Data</b> | numeric, not applicable |
| <b>Default Value</b>           | 0                       |
| <b>Minimum and Maximum</b>     | 0, 20                   |
| <b>Override</b>                | none                    |
| <b>Dependencies</b>            | none                    |
| <b>Recommendation</b>          | none                    |

## For Cache Refresh Process

The `/AgentSetup/AgentTuning/procCachePriority` configuration variable specifies the operating system scheduling priority for the process that refreshes the process cache.

|                                |                         |
|--------------------------------|-------------------------|
| <b>Format and Type of Data</b> | numeric, not applicable |
| <b>Default Value</b>           | 10                      |
| <b>Minimum and Maximum</b>     | 0, 20                   |
| <b>Override</b>                | none                    |
| <b>Dependencies</b>            | none                    |
| <b>Recommendation</b>          | none                    |

## For Internal Cache Refresh Process

The `/AgentSetup/AgentTuning/procCacheSchedPriority` configuration variable specifies the Internal PATROL Agent scheduling priority for the process that refreshes the process cache.

|                                |   |
|--------------------------------|---|
| <b>Format and Type of Data</b> | numeric, not applicable   |
| <b>Default Value</b>           | 1   |
| <b>Minimum and Maximum</b>     | 0, 20   |
| <b>Dependencies</b>            | none  |
| <b>Recommendation</b>          | Set this process high (0, 1, or 2). The PATROL process cache should be refreshed before other internal agent processes run. |

# Load Balancing the PATROL Agent

Load balancing the PATROL Agent involves setting the agent's internal run queue and managing which applications and instances are monitored and which parameters are run. This section describes how to set the Agent's internal run queue. For information on how to manage applications, see Chapter 7, "Loading and Monitoring Applications."

## PATROL Agent's Scheduling Policy

The `/AgentSetup/AgentTuning/runqSchedPolicy` configuration variable specifies the PATROL Agent's scheduling policy for the main run queue.

|                            |  |
|----------------------------|--|
|                            | The scheduling policy can be the sum of the following values:<br>1—next execution = finish time + poll time<br>2—next execution = last execution time + poll time<br>4—recalculate optimal scheduling upon each completion<br>8—force an interval of runqDelta |
| <b>Values</b>              |  |
| <b>Default Value</b>       | 1  |
| <b>Minimum and Maximum</b> | 1, 15  |

|                       |   |
|-----------------------|---|
| <b>Dependencies</b>   | none  |
| <b>Recommendation</b> | <p>If the value is set to <b>1</b> or <b>2</b>, the scheduling algorithm forces a spacing of runqDelta seconds between execution times of jobs in the Main Run Queue. The runqMaxDelta represents the maximum number of seconds that a job will be delayed from its “ideal” execution time.</p> <p>If the value is set to <b>4</b>, the agent attempts to find the “optimal” execution time by scheduling execution at a time that is furthest from its neighbors. The agent recalculates the runq slot after each execution. The calculation consumes a lot of resources.</p> <p>If the value is set to <b>8</b>, the agent schedules consecutive jobs at least runqDelta apart and runqMaxDelta does not apply. This option literally forces an interval of runqDelta between execution times. It affects scheduling of all jobs in the agent's Main run Queue and can allow for an indefinite delay in scheduling.</p> <p>This force interval option (<b>8</b>) should be used only after other measures such as changing poll times of various parameters have failed to reduce agent load.</p> |

## Interval Between Internal Processes

The `/AgentSetup/AgentTuning/runqDelta` configuration variable specifies the gap (in seconds) between processes in the PATROL Agent's run queue.

|                                |                  |
|--------------------------------|------------------|
| <b>Format and Type of Data</b> | numeric, seconds |
| <b>Default Value</b>           | 8                |
| <b>Minimum and Maximum</b>     | 1, none          |
| <b>Dependencies</b>            | none             |
| <b>Recommendation</b>          | none             |

## Increment Checking of Internal Process Interval

The `/AgentSetup/AgentTuning/runqDeltaIncrement` configuration variable specifies the increment (in seconds) used when checking for a gap in the PATROL Agent's run queue.

|                                |   |
|--------------------------------|---|
| <b>Format and Type of Data</b> | numeric, seconds  |
| <b>Default Value</b>           | 2   |
| <b>Minimum and Maximum</b>     | 1, must be less than or equal to the <code>runqDelta</code> |
| <b>Override</b>                | none  |
| <b>Dependencies</b>            | none  |
| <b>Recommendation</b>          | none  |

## Maximum Delay of an Internal Process

The `/AgentSetup/AgentTuning/runqMaxDelta` configuration variable specifies the maximum delay (in seconds) for a process. The `runqMaxDelta` represents the maximum number of seconds that a job will be delayed from its “ideal” execution time.

|                                |                  |
|--------------------------------|------------------|
| <b>Format and Type of Data</b> | numeric, seconds |
| <b>Default Value</b>           | 40               |
| <b>Minimum and Maximum</b>     | 10, none         |
| <b>Override</b>                | none             |
| <b>Dependencies</b>            | none             |
| <b>Recommendation</b>          | none             |

# Controlling the Number of Processes

The PATROL Agent allows you to limit the number of processes that are running simultaneously and the number of files that are open at the same time.

## Limit the Number of Files Opened Simultaneously

The PATROL Agent enables you to limit the number of files (and thus on Unix also the number of processes) simultaneously opened by the agent. It tracks files based on file descriptors. The maximum number of file descriptors depends on the machine resources and the behavior of process limit functions. The variable applies to both the console and agent.

The **PATROL\_MAX\_FILE\_DESCRIPTOR** environment variable specifies the maximum number of file descriptors that a system can have open at once. The maximum number of processes is determined by the platform.

|                                |  |
|--------------------------------|--|
| <b>Format and Type of Data</b> | numeric, file descriptors                    |
| <b>Default Value</b>           | 1024   |
| <b>Minimum and Maximum</b>     | 1, determined by OS and hardware limitations |
| <b>Dependencies</b>            | none   |
| <b>Recommendation</b>          | none   |

## Limit the Number of Processes Run Simultaneously

The `/AgentSetup/maxProcessLimit` configuration variable specifies the maximum number of processes that the agent can create and run simultaneously. The maximum number of processes is determined by the platform.

|                                |  |
|--------------------------------|--|
| <b>Format and Type of Data</b> | numeric, processes                           |
| <b>Default Value</b>           | 50   |
| <b>Minimum and Maximum</b>     | 1, determined by OS and hardware limitations |
| <b>Dependencies</b>            | none   |
| <b>Recommendation</b>          | none   |

## Limiting the Number of patrolperf Instances

You can use the `/AgentSetup/PerfMaxRestartCount` configuration variable to limit the number of `patrolperf` instances the PATROL Agent can spawn. The default value is 25. If `patrolperf` does not respond to the PATROL Agent, then the agent tries to spawn `patrolperf` again. The agent cannot spawn more than the number of instances defined in `PerfMaxRestartCount`.

|                                |                    |
|--------------------------------|--------------------|
| <b>Format and Type of Data</b> | numeric, processes |
| <b>Default Value</b>           | 25                 |
| <b>Minimum and Maximum</b>     | none               |
| <b>Dependencies</b>            | none               |
| <b>Recommendation</b>          | none               |

# Selecting PSL Debugging Level

The agent allows you to select the level of PSL runtime error checking messages that you want written to the PATROL Console's system output window.

## Set PSL Runtime Error Level

The `/AgentSetup/PslDebug` configuration variable determines the type and amount of runtime error messages sent to the system output window. If a console is not connected to the agent, the PSL runtime error messages are not generated or recorded.

|                            |  |
|----------------------------|--|
| <b>Values</b>              | The runtime error levels are<br><b>0</b> —None<br><b>1</b> —Full<br><b>32</b> —Only serious errors |
| <b>Default Value</b>       | 32   |
| <b>Minimum and Maximum</b> | not applicable   |
| <b>Dependencies</b>        | none   |
| <b>Recommendation</b>      | none   |

# Managing Performance Retrieval Parameters

The agent allows you to manage certain aspects of how it gathers performance data on Windows systems. You can determine

- the number of data values collected by the persistent data collection program (PatrolPerf.exe) before it restarts
- how long the agent waits for a response from the persistent data collection program
- where the data collection program writes debug information

---

## Note

---

These parameters only apply to the PATROL Agent running on Windows systems.

---

## Establishing a Timeout Period

The `/AgentSetup/PerfGetTimeout` agent configuration variable determines how long the agent waits for a response from the persistent data collection program.

|                                |                       |
|--------------------------------|-----------------------|
| <b>Format and Type of Data</b> | numeric, milliseconds |
| <b>Default Value</b>           | 10000                 |
| <b>Minimum and Maximum</b>     | not applicable        |
| <b>Dependencies</b>            | none                  |
| <b>Recommendation</b>          | none                  |

## Determining the Maximum Number of Data Points Collected Before Restarting

The `/AgentSetup/PerfGetMaxTimes` agent configuration variable determines the number of data points that the persistent data collection program collects before it restarts itself. Zero (0) indicates that the program does not restart itself.

|                                |                      |
|--------------------------------|----------------------|
| <b>Format and Type of Data</b> | numeric, data points |
| <b>Default Value</b>           | 0 (do not restart)   |
| <b>Minimum and Maximum</b>     | not applicable       |
| <b>Dependencies</b>            | none                 |
| <b>Recommendation</b>          | none                 |

## Specifying the Debug File

The `/AgentSetup/PerfGetDebugFile` agent configuration variable specifies the file to which the persistent data collection program writes debug information. The agent creates this file only if a “serious” error occurs. The debug level, and thus the definition of “serious,” is set internally. You cannot alter it.

|                                |   |
|--------------------------------|---|
| <b>Format and Type of Data</b> | text string, path and filename          |
| <b>Default Value</b>           | " " (empty string = do not create file) |
| <b>Minimum and Maximum</b>     | not applicable                          |
| <b>Dependencies</b>            | none                                    |
| <b>Recommendation</b>          | none                                    |

# Overriding Settings During a Session

The PATROL Console enables you to manage certain aspects of the PATROL Agent through menu commands and built-in commands. These management features temporarily override the values of agent configuration variables for the current session.

## Resetting Values

Once the PATROL Agent is shutdown and restarted, the PATROL Agent reloads the values stored in the agent configuration variables.

## Managing the PATROL Agent from the PATROL Console

For information on how to manage certain aspects of the PATROL Agent from the PATROL Console during a session, see Chapter 5, “Managing the PATROL Agent” in *PATROL<sup>®</sup> Console for Windows 2000 User Guide—Monitoring and Managing with PATROL (Volume 2)* or *PATROL<sup>®</sup> Console for Unix User Guide*.

---

## Glossary

### **access control list**

A list that is set up by using a PATROL Agent configuration variable and that restricts PATROL Console access to a PATROL Agent. A PATROL Console can be assigned access rights to perform console, agent configuration, or event manager activities. The console server uses access control lists to restrict access to objects in the COS namespace.

### **agent namespace**

*See* PATROL Agent namespace.

### **Agent Query**

A PATROL Console feature that constructs SQL-like statements for querying PATROL Agents connected to the console. Agent Query produces a tabular report that contains information about requested objects and can be used to perform object management activities, such as disconnecting and reconnecting computers. Queries can be saved, reissued, added, or changed. PATROL offers built-in queries in the Quick Query command on the Tools menu from the PATROL Console main menu bar. *See also* Quick Query.

### **alarm**

An indication that a parameter for an object has returned a value within the alarm range or that application discovery has discovered that a file or process is missing since the last application check. An alarm state for an object can be indicated by a flashing icon, depending on the configuration of a console preference. *See also* warning.

|                                |  |
|--------------------------------|--|
| <b>alert range</b>             | A range of values that serve as thresholds for a warning state or an alarm state. Alert range values cannot fall outside of set border range values. <i>See also</i> border action, border range, and recovery action.   |
| <b>ALL_ COMPUTERS class</b>    | The highest-level computer class in PATROL. Attributes assigned to this class will be inherited by all computer classes known to PATROL. <i>See also</i> class and computer class.   |
| <b>annotated data point</b>    | A specially marked point on a parameter graph that provides detailed information about a parameter at a particular moment. The associated data is accessed by double-clicking the data point, which is represented by a user-specified character (the default is an asterisk). <i>See also</i> parameter.  |
| <b>application account</b>     | An account that you define at KM setup and that you can change for an application class or instance. An application account is commonly used to connect to an RDBMS on a server where the database resides or to run SQL commands.   |
| <b>application check cycle</b> | The interval at which application discovery occurs. The PATROL Agent process cache (as opposed to the system process table) is checked to ensure that all application instances and files previously discovered still exist there. <i>See also</i> application discovery, application discovery rules, prediscovery, process cache refresh, PSL discovery, and simple discovery. |
| <b>application class</b>       | The object class to which an application instance belongs; also, the representation of the class as a container (Unix) or folder (Windows) on the PATROL Console. You can use the developer functionality of a PATROL Console to add or change application classes. <i>See also</i> class.   |

**application discovery**

A PATROL Agent procedure carried out at preset intervals on each monitored computer to discover application instances. When an instance is discovered, an icon appears on the PATROL interface. The application class includes rules for discovering processes and files by using simple process and file matching or PSL commands. Application definition information is checked against the information in the PATROL Agent process cache, which is periodically updated. Each time the PATROL Agent process cache is refreshed, application discovery is triggered. *See also* application check cycle, application discovery rules, PATROL Agent process cache, prediscovery, simple discovery, and PSL discovery.

**application discovery rules**

A set of rules stored by the PATROL Agent and periodically evaluated to find out whether a specific instance of an application class exists in the monitored environment. The rules describe how a PATROL Agent can detect instances of the application on a computer. There are two types of application discovery: simple and PSL; PSL discovery can include prediscovery rules as well as discovery rules. *See also* application check cycle, application discovery, simple discovery, prediscovery, and PSL discovery.

**application filter**

A feature used from the PATROL Console to hide all instances of selected application classes for a particular computer. The PATROL Agent continues to monitor the application instances by running parameter commands and recovery actions.

**application instance**

A system resource that is discovered by PATROL and that contains the information and attributes of the application class that it belongs to. *See also* application class and instance.

**application state**

The condition of an application class or an application instance. The most common application states are OK, warning, and alarm. An application class or instance icon can also show additional conditions. *See also* computer state and parameter state.

**attribute**

A characteristic that is assigned to a PATROL object (computer class, computer instance, application class, application instance, or parameter) and that you can use to monitor and manage that object. Computers and applications can have attributes such as command type, parameter, menu command, InfoBox command, PATROL setup command, state change action, or environment variable. Parameters can have attributes such as scheduling, command type, and thresholds.

An attribute can be defined globally for all instances of a class or locally for a particular computer or application instance. An instance inherits attributes from a class; however, an attribute defined at the instance level overrides inherited attributes. *See also* global level and local level.

**border action**

A command or recovery action associated with a parameter border range and initiated when that range has been breached. Border actions can be initiated immediately when the parameter returns a value outside the border range, after a warning or alarm has occurred a specified number of times, or after all other recovery actions have failed. *See also* border range.

**border range**

A range of values that serve as thresholds for a third-level alert condition when it is possible for a parameter to return a value outside of the alarm range limits. When a border range is breached, border actions can be initiated. *See also* border action.

**built-in command**

An internal command available from the PATROL Agent that monitors and manages functions such as resetting the state of an object, refreshing parameters, and echoing text. The command is identified by the naming convention `%command_name`. *See also* built-in macro variable.

**built-in macro variable**

An internal variable created and maintained by PATROL for use in built-in commands and PSL. The variable is identified by the naming convention `%{variable_name}`. *See also* built-in command.

|                               |  |
|-------------------------------|--|
| <b>chart</b>                  | <i>A plot of parameter data values made by the PATROL Console Charting Server. See multigraph container and PATROL Console Charting Server.</i>  |
| <b>charting server</b>        | <i>See PATROL Console Charting Server.</i>   |
| <b>class</b>                  | The object classification in PATROL where global attributes can be defined; the attributes are then inherited by instances of the class. An instance belongs to a computer class or an application class. <i>See also</i> application class, computer class, and event class.  |
| <b>collector parameter</b>    | A type of parameter that contains instructions for gathering values for consumer parameters to display. A collector parameter does not display any value, issue alarms, or launch recovery actions. <i>See also</i> consumer parameter, parameter, and standard parameter.   |
| <b>command line argument</b>  | An option for starting a PATROL Agent or a PATROL Console at the operating system command line. PATROL Agent arguments include names of KMs to load and port numbers for agent-console connection. PATROL Console arguments include connection mode (developer or operator), user ID to start the PATROL Console, names of KMs to load, and names of the files to use. |
| <b>command line interface</b> | <i>See PATROL Command Line Interface (CLI).</i>  |
| <b>command text editor</b>    | The component that provides basic text editing functions for a PATROL Console. It is commonly used to add or change commands (menu commands, parameter data collection and recovery actions, InfoBox commands, setup commands, and state change actions).  |

|   |  |
|---|--|
| <b>command type</b>                     | The designation assigned to a command according to its manner of execution. This attribute must be defined for a parameter command, a parameter recovery action, a menu command, an InfoBox command, a setup command, or a state change action. The PATROL Agent provides two command types: operating system (OS) and PSL. PATROL KMs provide additional command types. The developer functionality of a PATROL Console can be used to add or change command types. |
| <b>commit</b>                           | The process of saving to PATROL Agent computers the changes that have been made to a KM by using a PATROL Console. A PATROL user can disable a PATROL Console's ability to commit KM changes.  |
| <b>computer class</b>                   | The basic object class to which computer instances of the same type belong. Examples include Solaris, OSF1, HP, and RS6000. PATROL provides computer classes for all supported computers and operating systems; a PATROL Console with developer functionality can add or change computer classes.  |
| <b>computer instance</b>                | A computer that is running in an environment managed by PATROL and that is represented by an icon on the PATROL interface. A computer instance contains the information and attributes of the computer class that it belongs to. <i>See also</i> instance.   |
| <b>computer state</b>                   | The condition of a computer. The main computer states are OK, warning, and alarm. A computer icon can show additional conditions that include no output messages pending, output messages pending, void because a connection cannot be established, and void because a connection was previously established but now is broken. <i>See also</i> state.   |
| <b>configuration file, KM</b>           | <i>See</i> KM configuration file.  |
| <b>configuration file, PATROL Agent</b> | <i>See</i> PATROL Agent configuration file.  |

|                           |   |
|---------------------------|---|
| <b>connection mode</b>    | The mode in which the PATROL Console is connected to the PATROL Agent. The mode can be developer or operator and is a property of the Add Host dialog box (PATROL 3.x and earlier), an Add Managed System wizard, or other connecting method. The connection mode is a global (console-wide) property that can be overridden for a computer instance. <i>See also</i> PATROL Console.   |
| <b>console module</b>     | A program that extends the functionality of PATROL Central and PATROL Web Central. Console modules can collect data, subscribe to events, access Knowledge Module functions, authenticate users, and perform security-related functions. Console modules were formerly referred to as add-ons or snap-ins.  |
| <b>console server</b>     | A server through which PATROL Central and PATROL Web Central communicate with managed systems. A console server handles requests, events, data, communications, views, customizations, and security.  |
| <b>consumer parameter</b> | A type of parameter that displays a value that was gathered by a collector parameter. A consumer parameter never issues commands and is not scheduled for execution; however, it has alarm definitions and can run recovery actions. <i>See also</i> collector parameter, parameter, and standard parameter.  |
| <b>container</b>          | A custom object that you can create to hold any other objects that you select—such as computers, applications, and parameters—in a distributed environment. In Windows, a container is referred to as a folder. You can drag and drop an object into and out of a container icon. However, objects from one computer cannot be dropped inside another computer. Once a container is defined, the object hierarchy applies at each level of the container. That is, a container icon found within a container icon assumes the variable settings of the container in which it is displayed. <i>See also</i> object hierarchy and PATROL Console Charting Server. |
| <b>customize a KM</b>     | To modify properties or attributes locally or globally. <i>See also</i> global level and local level.   |

**custom view**

A grid-like view that can be created in PATROL Central or PATROL Web Central to show user-selected information.

**deactivate a parameter**

To stop running a parameter for selected computer or application instances. In PATROL Consoles for Microsoft Windows environments, deactivating a parameter stops parameter commands and recovery actions and deletes the parameter icon from the application instance window without deleting the parameter definition in the KM tree. A deactivated parameter can be reactivated at any time. *See also* snooze an alarm and suspend a parameter.

**deactivate an application class**

To stop monitoring an application class and all of its instances on selected computer instances. In PATROL Consoles for Microsoft Windows environments, deactivating an application class deletes the application class and all of its instance icons from the computer window without deleting the application class or definition in the KM tree. A deactivated application class can be reactivated at any time. *See also* application filter and deactivate a parameter.

**desktop file**

In PATROL 3.x and earlier, a file that stores your desktop layout, the computers that you monitor, the KMs that you loaded, and your PATROL Console user accounts for monitored objects. You can create multiple desktop files for any number of PATROL Consoles. By default, desktop files always have a **.dt** extension. Desktop files are replaced by management profiles in PATROL 7.x. *See also* desktop template file.

**desktop template file**

In PATROL 3.x and earlier, a file that stores information about the desktop setup of one computer. You can create multiple desktop template files for any number of PATROL Consoles. Each PATROL Console user can apply a template to selected computers on the desktop. By default, desktop template files always have a **.dtm** extension. *See also* desktop file.

**Desktop tree**

*A feature of PATROL for Microsoft Windows only.* One of the views of folders available with PATROL for Microsoft Windows environments, the Desktop tree displays the object hierarchy. *See also* KM tree.

**developer mode**

An operational mode of the PATROL Console that can be used to monitor and manage computer instances and application instances and to customize, create, and delete locally loaded Knowledge Modules and commit these changes to selected PATROL Agent computers. *See* PATROL Console.

**disable an application, disable a KM**

To temporarily or permanently block an application or KM from loading and to block the PATROL Agent from using that KM. When a KM is disabled (added to the disabled list) in the agent configuration file, the KM files are not deleted from the PATROL Agent computers, but the PATROL Agent stops using the KM to collect parameter data and run recovery actions. The default is that no KMs are disabled. Most KMs are composed of individual application files with a **.km** extension. *See also* preloaded KM, static KM, and unload a KM.

**discovery**

*See* application discovery.

**distribution CD or tape**

A CD or tape that contains a copy of one or more BMC Software products and includes software and documentation (user guides and online help systems).

**environment variable**

A variable used to specify settings, such as the program search path for the environment in which PATROL runs. You can set environment variables for computer classes, computer instances, application classes, application instances, and parameters.

**event**

The occurrence of a change, such as the appearance of a task icon, the launch of a recovery action, the connection of a console to an agent, or a state change in a monitored object (computer class, computer instance, application class, application instance, or parameter). Events are captured by the PATROL Agent, stored in an event repository file, and forwarded to an event manager (PEM) if an event manager is connected. The types of events forwarded by the agent are governed by a persistent filter for each event manager connected to a PATROL Agent.

**event acknowledgment command**

A command that is triggered by the PATROL Agent when an event is acknowledged in an event manager (PEM). *See also* event escalation command and event notification command.

|                                 |   |
|---------------------------------|---|
| <b>event catalog</b>            | A collection of event classes associated with a particular application. PATROL provides a Standard Event Catalog that contains predefined Standard Event Classes for all computer classes and application classes. You can add, customize, and delete an application event catalog only from a PATROL Console in the developer mode. <i>See also</i> event class and Standard Event Catalog.  |
| <b>event class</b>              | A category of events that you can create according to how you want the events to be handled by an event manager and what actions you want to be taken when the event occurs. Event classes are stored in event catalogs and can be added, modified, or deleted only from a PATROL Console in the developer mode. PATROL provides a number of event classes in the Standard Event Catalog, such as worst application and registered application. <i>See also</i> event catalog and Standard Event Catalog. |
| <b>event class command</b>      | A command that is run by the PATROL Agent when certain events occur and that is used in conjunction with an event manager (PEM). The commands are specified for the event class that the event is associated with. A command can be one of three types: escalation, notification, or acknowledgment. <i>See also</i> event acknowledgment command, event escalation command, and event notification command.  |
| <b>Event Diary</b>              | The part of an event manager (PEM) where you can store or change comments about any event in the event log. You can enter commands at any time from the PATROL Event Manager Details window.  |
| <b>event escalation command</b> | A command that is triggered by the PATROL Agent when an event is not acknowledged, closed, or deleted within an event manager (PEM) by the end of the escalation period. <i>See also</i> event acknowledgment command, event escalation period, and event notification command.   |

**event escalation period**

A period during which an increase in the severity of an event occurs as the result of its persistence. Escalation actions are defined as part of escalation command definitions for event classes and can be triggered only by the PATROL Agent. *See also* event escalation command.

**event history repository**

A circular file where events are stored by the PATROL Agent and accessed by an event manager, such as the PEM. The file resides on the PATROL Agent computer and retains a limited number of events. When the maximum number of events is reached and a new event is stored, the oldest event is removed in a cyclical fashion. *See also* parameter history repository.

**event manager**

A graphical user interface for monitoring and managing events. The event manager can be used with or without the PATROL Console. *See also* PATROL Event Manager (PEM).

**event notification command**

A command that is triggered by the PATROL Agent when an event is logged into an event manager (PEM). *See also* event acknowledgment command and event escalation command.

**event type**

The PATROL-provided category for an event according to a filtering mechanism in an event manager. Event types include information, state change, error, warning, alarm, and response.

**event view filter**

*See* view filter.

**event-driven scheduling**

A kind of scheduling that starts a parameter when certain conditions are met. *See also* periodic scheduling.

**expert advice**

Comments about or instructions for dealing with PATROL events as reported by the agent. Expert advice is defined in the Event Properties dialog box in a PATROL Console in the developer mode. PATROL Consoles in an operator mode view expert advice in the PATROL Event Manager.

**filter, application**

*See* application filter.

**filter, event view**

*See* view filter.

|                           |  |
|---------------------------|--|
| <b>filter, persistent</b> | <i>See</i> persistent filter.  |
| <b>global channel</b>     | A single dedicated connection through which PATROL monitors and manages a specific program or operating system. The PATROL Agent maintains this connection to minimize the consumption of program or operating system resources.   |
| <b>global level</b>       | In PATROL hierarchy, the level at which object properties and attributes are defined for all instances of an object or class. An object at the local level inherits characteristics (properties) and attributes from the global level. <i>See also</i> local level.  |
| <b>heartbeat</b>          | A periodic message sent between communicating objects to inform each object that the other is still “alive.” For example, the PATROL Console checks to see whether the PATROL Agent is still running.  |
| <b>heartbeat interval</b> | The interval (in seconds) at which heartbeat messages are sent. The longer the interval, the lower the network traffic. <i>See also</i> message retries, message time-out, and reconnect polling.  |
| <b>history</b>            | Parameter and event values that are collected and stored on each monitored computer. Parameter values are stored in binary files for a specified period of time; events are stored in circular log files until the maximum size is reached. The size and location of parameter history files are specified through either the PATROL Console or the PATROL Agent; size and location of event history files are specified through an event manager, such as the PEM, or the PATROL Agent. |
| <b>history repository</b> | A binary file in which parameter values (except those that are displayed as text) are stored by the PATROL Agent and accessed by the PATROL Console for a specified number of days (the default is one day). When the number of storage days is reached, those values are removed in a cyclical fashion.   |

|                                 |   |
|---------------------------------|---|
| <b>history retention level</b>  | The specified level (global or local) where the parameter history retention period for an object is set. The period can be inherited from the next higher level in the object hierarchy or set at the local level. If the history retention level is local, the number of days that history is stored (retention period) must be set. <i>See also</i> history retention period.   |
| <b>history retention period</b> | The number of days that parameter values are stored in the history database before they are automatically purged by PATROL. The period can be specified at the class (global) or instance (local) level. History retention can be set for all parameters of a computer class, a computer instance, an application class, or an application instance. History for an individual parameter on an application instance can be manually cleared at any time by using a PATROL Console. <i>See also</i> history retention level. |
| <b>history span</b>             | The combined settings for a parameter history retention level and period. <i>See also</i> history retention level and history retention period.   |
| <b>InfoBox</b>                  | A dialog box that contains a static list of fields and displays current information about an object, such as the version number of an RDBMS and whether the object is online or offline. Commands are run when the InfoBox is opened. Information can be manually updated if the InfoBox remains open for a period of time. PATROL provides a number of commands for obtaining and displaying object information in an InfoBox. Only a PATROL Console in the developer mode can be used to add or change commands.          |
| <b>information event</b>        | Any event that is not a state change or an error. Typical information events occur when a parameter is activated or deactivated, a parameter is suspended or resumed, or application discovery is run. The default setting for PATROL is to prevent this type of event from being stored in the event repository. To store and display this type of event, you must modify the persistent filter setting in the PATROL Agent configuration file.  |

|                              |  |
|------------------------------|--|
| <b>instance</b>              | A computer or discovered application that is running in an environment managed by PATROL. An instance has all the attributes of the class that it belongs to. A computer instance is a monitored computer that has been added to the PATROL Console. An application instance is discovered by PATROL. See application discovery, application instance, and computer instance.  |
| <b>KM</b>                    | <i>See</i> Knowledge Module (KM).  |
| <b>KM configuration file</b> | A file in which the characteristics of a KM are defined through KM menu commands during KM installation and setup (if setup is required). <i>See also</i> Knowledge Module (KM) and PATROL Agent configuration file.   |
| <b>KM list</b>               | A list of KMs used by a PATROL Agent or PATROL Console. <i>See also</i> Knowledge Module (KM).   |
| <b>KM Migrator</b>           | <i>See</i> PATROL KM Migrator and Knowledge Module (KM).   |
| <b>KM package</b>            | <i>See</i> Knowledge Module package.   |
| <b>KM tree</b>               | <i>A feature of PATROL for Microsoft Windows only.</i> One of two views of folders available in Windows. The KM tree displays computer classes, application classes, and their customized instances in the knowledge hierarchy and also displays the Standard Event Catalog. A PATROL Console in operator mode can only view the KM tree; only a PATROL Console in the developer mode can change KM properties and attributes. <i>See also</i> Desktop tree and Knowledge Module (KM). |
| <b>knowledge hierarchy</b>   | The rules by which objects inherit or are assigned attributes. (In PATROL Consoles for Microsoft Windows environments, classes of objects are represented in the Computer Classes and Application Classes sets of folders on the KM tree.) Properties and attributes of a customized instance override those defined for the class to which the instance belongs.  |

**Knowledge Module (KM)**

A set of files from which a PATROL Agent receives information about resources running on a monitored computer. A KM file can contain the actual instructions for monitoring objects or simply a list of KMs to load. KMs are loaded by a PATROL Agent and a PATROL Console.

KMs provide information for the way monitored computers are represented in the PATROL interface, for the discovery of application instances and the way they are represented, for parameters that are run under those applications, and for the options available on object pop-up menus. A PATROL Console in the developer mode can change KM knowledge for its current session, save knowledge for all of its future sessions, and commit KM changes to specified PATROL Agent computers. *See also commit, KM configuration file, KM list, KM Migrator, KM tree, load KMs, and version arbitration.*

**Knowledge Module package**

A package of PATROL KM files that can be distributed by an installation program or stored in and distributed by the PATROL KMDS. The package file has a **.pkg** file extension. KM packages are created by using a PATROL Console in the developer mode. *See also Knowledge Module (KM), PATROL Console, PATROL Knowledge Module Deployment Server (PATROL KMDS), and PATROL Knowledge Module Deployment Server Manager (PATROL KMDS Manager).*

**load applications**

Same as load KMs. Most KMs are composed of application files with a **.km** extension.

**load KMs**

To place KM files into memory for execution. After configuration and during startup, the PATROL Agent loads the KM files that are listed in its configuration file and that reside on the PATROL Agent computer. When a PATROL Console connects to the PATROL Agent, the KM versions that the agent executes depend on whether the console has developer or operator functionality. *See also Knowledge Module (KM) and version arbitration.*

**local history**

The history (stored parameter values) for an object or instance. *See also global level and local level.*

|                                       |   |
|---------------------------------------|---|
| <b>local history retention period</b> | The length of time set by the user during which stored parameter values for an object or instance are retained.   |
| <b>local level</b>                    | In PATROL hierarchy, the level of a computer instance or an application instance. An object (instance) at the local level inherits properties and attributes that are defined globally. When properties and attributes are customized locally for an individual instance, they override inherited attributes. <i>See also</i> global level.   |
| <b>managed object</b>                 | Any object that PATROL manages. <i>See</i> object.  |
| <b>managed system</b>                 | A system—usually a computer on which a PATROL Agent is running—that is added (connected) to a PATROL Console to be monitored and managed by PATROL and that is represented by an icon on the PATROL interface.  |
| <b>management profile</b>             | A user profile for PATROL Central and PATROL Web Central that is stored by the console server. A management profile is similar to a session file and contains information about custom views, your current view of the PATROL environment, information about systems that you are currently managing, Knowledge Module information, and console layout information for PATROL Central. Management profiles replace desktop files and session files that were used in PATROL 3.x and earlier.                          |
| <b>master agent</b>                   | <i>See</i> PATROL SNMP Master Agent.  |
| <b>message retries</b>                | <i>A feature of UDP only.</i> The number of times that the PATROL Console will resend a message to the PATROL Agent. The greater the number of message retries, the more time the PATROL Console will give the PATROL Agent to respond before deciding that the agent connection is down and timing out. The number of message retries multiplied by message time-out (in seconds) is the approximate time allowed for a connection verification. <i>See also</i> heartbeat, message time-out, and reconnect polling. |

|                             |  |
|-----------------------------|--|
| <b>message time-out</b>     | <i>A feature of UDP only.</i> The time interval (in seconds) that the PATROL Console will give the PATROL Agent to respond to a connection verification before deciding that the Agent connection is down. The number of message retries multiplied by message time-out is the approximate time allowed for a connection verification. <i>See also</i> heartbeat, message retries, and reconnect polling.  |
| <b>message window</b>       | A window that displays command output and error messages from the PATROL Console graphical user interface. <i>See also</i> response window, system output window, and task output window.  |
| <b>multigraph container</b> | A custom object into which you can drop parameter objects to be plotted as charts. <i>See also</i> PATROL Console Charting Server.   |
| <b>object</b>               | A computer class, computer instance, application class, application instance, parameter, or container (folder) in an environment managed by PATROL. Objects have properties and are assigned attributes (command types, parameters, menu commands, InfoBox commands, setup commands, state change actions, and environment variables). Parameter objects use data collection commands to obtain values from classes and instances. <i>See also</i> object class, object hierarchy, object icon, and object window. |
| <b>object class</b>         | A computer class or application class. <i>See also</i> class, object, and object hierarchy.  |
| <b>object hierarchy</b>     | The structure of object levels in PATROL. On the PATROL interface, computers contain application folders (containers) representing a loaded KM, application folders contain one or more application instances, and application instances contain parameters.   |
| <b>object icon</b>          | A graphic that represents a computer instance, application class, application instance, parameter, or container (folder) in an environment managed by PATROL. <i>See also</i> object, object hierarchy, and object window.   |

**object window**

An open object container (folder) that may contain application class icons, application instance icons, parameter icons, custom containers (folders), and shortcuts. The object window is displayed when you double-click the object icon. *See also* application instance, computer instance, object, and object icon.

**operator mode**

An operational mode of the PATROL Console that can be used to monitor and manage computer instances and application instances but not to customize or create KMs, commands, and parameters. *See* PATROL Console.

**operating system account**

An account that is set up at installation to grant the PATROL Agent access to a computer. Operating system commands executed by the PATROL Agent and PATROL Console use this account. The PATROL Agent configuration specifies a default operating system account, which can be changed.

**override a parameter**

To disable or change the behavior of a local PATROL application parameter. The changes to the parameter are local to the managed system running the parameter and are stored in the agent configuration database. You must be granted specific permissions by a PATROL Administrator through the PATROL User Roles file in order to override parameters. *See also* PATROL roles.

**parameter**

The monitoring element of PATROL. Parameters are run by the PATROL Agent; they periodically use data collection commands to obtain data on a system resource and then parse, process, and store that data on the computer that is running the PATROL Agent. Parameters can display data in various formats, such as numeric, text, stoplight, and Boolean. Parameter data can be accessed from a PATROL Console, PATROL Integration products, or an SNMP console. Parameters have thresholds and can trigger warnings and alarms. If the value returned by the parameter triggers a warning or an alarm, the PATROL Agent notifies the PATROL Console and runs any recovery actions associated with the parameter. *See also* parameter history repository and parameter state.

|  |   |
|--|---|
| <b>parameter cache</b>                     | The memory location where current parameter data is kept. In the PATROL Agent's configuration file, you can set the size of the cache, the maximum number of data points that can be stored, and the interval (in seconds) for emptying the cache.  |
| <b>parameter history repository</b>        | Also known as parameter history file. <i>See</i> history repository.  |
| <b>parameter override</b>                  | <i>See</i> override parameter.  |
| <b>parameter state</b>                     | The condition of a parameter. The most common parameter states are OK, warning, and alarm. A parameter icon can show additional conditions that include no history, offline, and suspended. A parameter can also be deactivated; when a parameter is deactivated, no icon is displayed. <i>See also</i> state.  |
| <b>PATROL Agent</b>                        | The core component of PATROL architecture. The agent is used to monitor and manage host computers and can communicate with the PATROL Console, a stand-alone event manager (PEM), PATROL Integration products, and SNMP consoles. From the command line, the PATROL Agent is configured by the pconfig utility; from a graphical user interface, it is configured by the xpconfig utility for Unix or the wpconfig utility for Windows. <i>See also</i> PATROL SNMP Master Agent. |
| <b>PATROL Agent configuration file</b>     | A file in which you can define the characteristics of the PATROL Agent by setting PATROL Agent configuration variables. You can edit the configuration file by using the pconfig utility, the wpconfig utility, or the xpconfig utility. <i>See also</i> KM configuration file, PATROL Agent configuration variable, pconfig, wpconfig, and xpconfig.   |
| <b>PATROL Agent configuration variable</b> | The means by which the characteristics of a PATROL Agent are defined. PATROL provides default variable values that can be customized. Configuration variables determine such characteristics as how errors are handled, which KMs are loaded and how, how SNMP support is configured, and how events trigger SNMP traps. <i>See also</i> PATROL Agent configuration file.   |
| <b>PATROL Agent Manager</b>                | <i>A feature of PATROL for Microsoft Windows only.</i> The graphical user interface used to install and run the PATROL Agent.   |

**PATROL Agent namespace**

A memory array that contains an internal representation of the PATROL object hierarchy. Values in the agent namespace are available to PSL scripts, eliminating the need to develop code to collect this data.

**PATROL Agent process cache**

A snapshot of the operating system process table on a monitored computer. The agent process cache is updated periodically.

**PATROL Agent process cache refresh**

A periodic process of the PATROL Agent that issues a platform-dependent system query to obtain a list of the active processes. This data is used to update the PATROL Agent process cache.

**PATROL Agent run queue**

*A time-ordered schedule of actions, such as application discovery and parameter execution, to be carried out by the PATROL Agent. See also PSL run queue.*

**PATROL Command Line Interface (CLI)**

An interface program that you can access from the command line of a monitored computer and through which you can run some PATROL products and utilities. With the CLI, you can monitor the state of PATROL Agents remotely, execute PSL functions, and query and control events. The CLI is used in place of the PATROL Console when memory and performance constraints exist.

## **PATROL Console**

The graphical user interface from which you launch commands and manage the environment monitored by PATROL. The PATROL Console displays all of the monitored computer instances and application instances as icons. It also interacts with the PATROL Agent and runs commands and tasks on each monitored computer. The dialog is event-driven so that messages reach the PATROL Console only when a specific event causes a state change on the monitored computer.

A PATROL Console with developer functionality can monitor and manage computer instances, application instances, and parameters; customize, create, and delete locally loaded Knowledge Modules and commit these changes to selected PATROL Agent computers; add, modify, or delete event classes and commands in the Standard Event Catalog; and define expert advice. A PATROL Console with operator functionality can monitor and manage computer instances, application instances, and parameters and can view expert advice but not customize or create KMs, commands, and parameters. *See also* developer mode and operator mode.

## **PATROL Console Charting Server**

A PATROL function that creates charts and graphs of actual values returned by more than one parameter. Charts and graphs are created by dragging and dropping various parameters into a multigraph container (folder) and plotting the results into a chart. Parameter data is plotted either in real time or from history sets and can be presented in a number of chart styles, including line graphs, pie charts, 3-D bar charts, and area plots. Charts can be viewed through the PATROL Console and printed to a local printer or PostScript file.

## **PATROL Enterprise Manager (PATROL EM)**

An event management system that gathers, filters, translates, and prioritizes messages from the managed systems in an enterprise and displays them as alerts in a single console. The PATROL EM consolidates alerts from different vendors and different geographical locations into a single display for fast identification and resolution of potential problems.

**PATROL Event Manager (PEM)**

An event manager that you can use to view and manage events that occur on monitored system resources and that are sent by PATROL Agents. You can access the PEM from the PATROL Console or use it as a stand-alone facility. It works with the PATROL Agent and user-specified filters to provide a customized view of events. *See also* event manager.

**PATROL History Loader KM**

A PATROL utility used to convert PATROL parameter history data into an ASCII data file or to store parameter history data directly into a particular relational database management system.

**PATROL Integration products**

Formerly PATROLVIEW or PATROLINK. Products that can be used to view events and to monitor and display all the parameters provided by the PATROL Agents and KMs in a network or enterprise management console.

**PATROL KMDS**

*See* PATROL Knowledge Module Deployment Server (PATROL KMDS).

**PATROL KMDS Manager**

*See* PATROL Knowledge Module Deployment Server Manager (PATROL KMDS Manager).

**PATROL KM Migrator**

A PATROL utility used to propagate KM user customizations to newly released versions of PATROL Knowledge Modules.

**PATROL Knowledge Module Deployment Server (PATROL KMDS)**

The change and version control tool for KMs. A repository for storage of PATROL KMs and changes to those KMs.

**PATROL Knowledge Module Deployment Server Manager (PATROL KMDS Manager)**

The graphical interface for the PATROL KMDS that can be used to manage and deploy or distribute KM changes in the production environment.

**PATROL roles**

A set of permissions that grant or remove the ability of a PATROL Console or PATROL Agent to perform certain functions. PATROL roles are defined in the PATROL User Roles file, which is read when the console starts.

**PATROL Script Language (PSL)**

A scripting language (similar to Java) that is used for generic system management and that is compiled and executed on a virtual machine running inside the PATROL Agent. PSL is used for writing application discovery procedures, parameters, recovery actions, commands, and tasks for monitored computers within the PATROL environment.

**PATROL SNMP Master Agent**

The agent through which a PATROL Agent interacts with an SNMP agent and SNMP manager. The PATROL Master Agent configuration file contains the community name and port number for all agents in such a multiple-agent architecture.

**patroldev**

A domain group that can be set up by a Windows system administrator to restrict user access to a PATROL Developer Console. When a user tries to start a PATROL Console with developer functionality, PATROL checks whether the user is in the patroldev group. If the user is not in the group, a PATROL Console with operator functionality is started instead. *See also* ptrldev.

**pconfig**

The command line utility for setting PATROL Agent configuration variables. *See also* PATROL Agent configuration file, PATROL Agent configuration variable, wpconfig, and xpconfig.

**PEM**

See PATROL Event Manager (PEM).

**periodic scheduling**

A kind of scheduling that starts a parameter at a certain time and reruns the parameter at certain intervals. *See also* event-driven scheduling.

**persistent filter**

A filter maintained by the PATROL Agent for each PATROL Console or event manager that connects to it. The filter is used to minimize network traffic by limiting the number and types of events that are forwarded from a PATROL Agent to a PATROL Console or an event manager (PEM).

**polling cycle**

The schedule on which a parameter starts running and the intervals at which it reruns; the cycle is expressed in seconds. *See also* event-driven scheduling and periodic scheduling.

|                              |  |
|------------------------------|--|
| <b>pop-up menu</b>           | The menu of commands for a monitored object; the menu is accessed by right-clicking the object.  |
| <b>prediscovery</b>          | A quick one-time test written in PSL to determine whether a resource that you want to monitor is installed or running on a monitored computer. If the results are affirmative, the PATROL Agent runs the discovery script. Prediscovery helps reduce PATROL Agent processing requirements.   |
| <b>preloaded KM</b>          | A KM that is loaded by the PATROL Agent at startup and run as long as the Agent runs. <i>See also</i> disable a KM and static KM.  |
| <b>process cache refresh</b> | <i>See</i> PATROL Agent process cache refresh.   |
| <b>property</b>              | A characteristic or attribute of an object, such as its icon.  |
| <b>PSL</b>                   | <i>See</i> PATROL Script Language (PSL).   |
| <b>PSL Compiler</b>          | A PATROL utility that compiles PSL scripts into a binary byte code that can be executed by the PSL virtual machine. The PSL Compiler can also be used to check a PSL script for syntax errors. The compiler is embedded in the PATROL Agent and PATROL Console (PATROL 3.x and earlier) and can also be run as a command-line utility. |
| <b>PSL Debugger</b>          | A PATROL Console utility that is used to debug PSL scripts. The PSL debugger is accessed through a computer's pop-up menu.   |
| <b>PSL discovery</b>         | A type of application discovery in which the discovery rules are defined by using PSL. PSL discovery can consist of prediscovery and discovery PSL scripts.  |
| <b>PSL Profiler</b>          | A PATROL utility that is used to tune the CPU usage and minimize child processes or file operations of a newly created KM. When the PSL Profiler is enabled, the PATROL Agent starts accumulating and recording profile statistics.  |
| <b>PSL run queue</b>         | A queue of the currently executing PSL processes. The PAL run queue is used to distribute processing time between PAL processes in a round-robin fashion.  |

|                          |  |
|--------------------------|--|
| <b>ptrldev</b>           | A form of patroldev that can be used in environments that support domain names no larger than eight characters. <i>See</i> patroldev.  |
| <b>Quick Query</b>       | In PATROL 3.x and earlier, a command on the Tools menu from the PATROL Console main menu bar that contains built-in predefined commands that you can use to query the agent for frequently needed information. For example, you can query the agent regularly about all computer instances, application instances, and parameters that are in a warning or alarm state. <i>See also</i> Agent Query. |
| <b>reconnect polling</b> | The time interval (in seconds) at which the PATROL Console will try to reconnect to a PATROL Agent that has dropped the previous connection. The longer the interval, the lower the network traffic. <i>See also</i> heartbeat, message retries, message time-out.   |
| <b>recovery action</b>   | A procedure that attempts to fix a problem that caused a warning or alarm condition. A recovery action is defined within a parameter by a user or by PATROL and triggered when the returned parameter value falls within a defined alarm range.  |
| <b>refresh parameter</b> | An action that forces the PATROL Agent to run one or more parameters immediately, regardless of their polling cycle. Refreshing does not reset the polling cycle but gathers a new data point between polling cycles.  |
| <b>reporting filter</b>  | The filter used by the PATROL Agent when transmitting events to consoles (event cache) from the event repository (located at the agent) for statistical reports.   |
| <b>response window</b>   | An input and output display for many KM menu commands that provides a customizable layout of the information (for example, the sort method for outputting system process IDs). <i>See also</i> system output window and task output window.  |
| <b>run queue</b>         | <i>See</i> PATROL Agent run queue.   |

|  |   |
|--|---|
| <b>self-polling parameter</b>                    | A standard parameter that starts a process that runs indefinitely. The started process periodically polls the resource that it is monitoring and emits a value that is captured by the PATROL Agent and published as the parameter value. Self-polling avoids the overhead of frequently starting external processes to collect a monitored value. A self-polling parameter differs from most other parameters that run scripts for a short time and then terminate until the next poll time. |
| <b>session file</b>                              | In PATROL 3.x and earlier, any of the files that are saved when changes are made and saved during the current PATROL Console session. A session file includes the <b>session-1.km</b> file, which contains changes to KMs loaded on your console, and the <b>session-1.prefs</b> file, which contains user preferences. Session files are replaced by management profiles in PATROL 7.x.  |
| <b>setup command</b>                             | A command that is initiated by the PATROL Console and run by the PATROL Agent when the PATROL Console connects or reconnects to the agent. For example, a setup command can initialize an application log file to prepare it for monitoring. PATROL provides some setup commands for computer classes. Only a PATROL Console with developer functionality can add or change setup commands.   |
| <b>shortcut</b>                                  | An alias or copy of an object icon in the PATROL hierarchy.   |
| <b>simple discovery</b>                          | A type of application discovery that uses simple pattern matching for identifying and monitoring files and processes.   |
| <b>SNMP</b>                                      | <i>See</i> Simple Network Management Protocol.  |
| <b>Simple Network Management Protocol (SNMP)</b> | A communications protocol that is supported by the PATROL Agent. SNMP allows network management systems to access PATROL Agents and allows PATROL Agents to monitor and manage SNMP devices.  |
| <b>SNMP trap</b>                                 | A condition which, when satisfied, results in an SNMP agent issuing a trap message to other SNMP agents and clients. Within the PATROL Agent, all events can be translated to SNMP traps and forwarded to SNMP managers.  |

**snooze an alarm**

To temporarily suspend an alarm so that a parameter does not exhibit an alarm state. During the user-set snooze period, the parameter continues to run commands and recovery actions, and the parameter icon appears to be in an OK state. *See also* deactivate a parameter and suspend a parameter.

**Standard Event Catalog**

A PATROL-provided collection of predefined event classes for all computer classes and application classes. To add, modify, or delete event classes and commands in the Standard Event Catalog, you must use a PATROL Console with developer functionality. *See also* event catalog and event class.

**standard parameter**

A type of parameter that collects and displays data and can also execute commands. A standard parameter is like a collector parameter and consumer parameter combined. *See also* collector parameter, consumer parameter, and parameter.

**startup command**

*See* setup command.

**state**

The condition of an object (computer instance, application instance, or parameter) monitored by PATROL. The most common states are OK, warning, and alarm. Object icons can show additional conditions. *See also* application state, computer state, parameter state, and state change action.

**state Boolean**

A parameter output style that represents the on or yes state of a monitored object as a check mark and the off or no state as the letter *x*. Parameters with this output style can have alerts (warning and alarm) and recovery actions. Numeric data output for the monitored object can be displayed as a graph. *See also* spotlight.

**state change action**

An action that is stored, maintained, and initiated by the PATROL Console when the console is notified by the PATROL Agent that a monitored object has changed state. The action, or command, executes on the computer on which the console is running, not the computer on which the agent is running.

**static KM**

A KM that is not loaded by the PATROL Agent before a PATROL Console with a loaded KM of the same name connects to the Agent. Once loaded by the agent, a static KM is never unloaded but continues to run as long as the agent runs, even if all PATROL Consoles with a registered interest disconnect from the PATROL Agent. If the PATROL Agent stops, static KMs will not be reloaded. *See also* disable a KM and preloaded KM.

**stoplight**

A parameter output style that displays OK, warning, and alarm states as green, yellow, and red lights, respectively, on a traffic light. Parameters with this output style can have alerts (warning and alarm) and recovery actions. Numeric data output for the monitored object can be displayed as a graph. *See also* state Boolean.

**suspend a parameter**

To stop running a parameter for selected computers or application instances. Suspending a parameter stops parameter commands and recovery actions but does not delete the parameter icon from the application instance window and does not delete the parameter definition from the KM tree in PATROL Consoles for Microsoft Windows environments. A suspended parameter can be resumed at any time. You can suspend a parameter from its pop-up menu. *See also* deactivate a parameter and snooze an alarm.

**system output window**

A message window that displays the output of commands and tasks that the PATROL Console or the PATROL Agent execute on an instance. The windows also displays error messages, commit status messages, and so forth. When the system output window contains unread messages, PATROL displays a yellow triangle for Windows; for Unix, it displays a blue screen with white text.

**task**

A command or group of commands that can execute on one object or several objects simultaneously. A task runs in the background and is not part of the PATROL Agent run queue; a task icon is displayed for each running task.

|                                     |   |
|-------------------------------------|---|
| <b>task output window</b>           | A window that contains command output generated by a task (for example, a KM menu command or a parameter warning or alarm). While executing, each task has its own icon, which usually appears in the PATROL interface or main window but may appear in an appropriate object window.   |
| <b>threshold</b>                    | A point or points that define a range of values, outside of which a parameter is considered to be in a warning or alarm range.  |
| <b>unload a KM</b>                  | To delete a KM from a PATROL Console session in order to stop monitoring the KM-defined objects on all computers. The KM files are not deleted from the directories on the PATROL Console or the PATROL Agent computers, and the PATROL Agent will continue to run the KM, collect parameter data, and run recovery actions until no connected console has the KM loaded. To prevent the PATROL Agent computer from collecting parameter data and running recovery actions for a KM, disable the KM. If a KM has been flagged as static, then it will not be unloaded. <i>See also</i> disable a KM, preloaded KM, and static KM. |
| <b>User Datagram Protocol (UDP)</b> | In PATROL 3.x and earlier, a connectionless network protocol that allows the PATROL Console to connect to many agents simultaneously. TCP requires an open file for each connection, and the number of files that a process may have open is generally limited.   |
| <b>user preferences</b>             | The PATROL Console settings that designate the account that you want to use to connect to monitored host computers, prevent a console with developer functionality from downloading its version of a KM to a PATROL Agent upon connection, disable the commit process for a console with developer functionality, determine certain window and icon display characteristics, specify the event cache size, and indicate whether startup and shutdown commands are enabled. A PATROL Console with either developer or operator functionality can change user preferences.  |

**version arbitration**

The KM version comparison that PATROL makes when a PATROL Console connects to a PATROL Agent. By default, KM versions from PATROL Consoles with developer functionality are loaded rather than PATROL Agent KM versions, and PATROL Agent KM versions are loaded rather than KM versions from PATROL Consoles with operator functionality.

**view filter**

A filter that can be created in an event manager (PEM) and that screens events forwarded from PATROL Agents. Views can be created, stored, and reapplied to host computers.

**warning**

An indication that a parameter has returned a value that falls within the warning range. *See also* alarm.

**wpconfig**

*A feature of PATROL for Microsoft Windows only.* The graphical user interface utility for setting PATROL Agent configuration variables. The wpconfig utility can be accessed from a computer pop-up menu on a computer running a PATROL Agent or a computer running a PATROL Console with developer functionality. *See also* PATROL Agent configuration file and PATROL Agent configuration variable.

**xpconfig**

*A feature of PATROL for Unix only.* The graphical user interface utility for setting PATROL Agent configuration variables. You can access the xpconfig utility from an xterm session command line on a computer running a PATROL Agent or from a pop-up menu or an xterm session command line on a PATROL Console with developer functionality. *See also* PATROL Agent configuration file and PATROL Agent configuration variable.

---

# Index

## A

- access control list
  - accounts 5-15
  - for the agent 5-9
  - format 5-9
  - host 5-13
  - modes 5-15
  - user 5-13
- Access Control List (ACL) 13-2
- account
  - default 4-2
  - for specific application 4-3
  - for specific instance 4-4
- ACPI E-2
- Add Hosts Dialog Box 9-34
- Add Variable Dialog Box 9-36, 10-12
- agent audit log 11-21
- Agent Query 1-8
- anchors
  - examples F-7
- application
  - status 7-6
- application class status 16-12
- application discovery
  - rate H-9
- application instances
  - displayed by the PATROL Agent 7-19

- applications
  - disabled 7-6
  - dynamic 7-6
  - filter list 7-19
  - loading status 7-6
  - monitoring 7-18
    - include or exclude 7-18
  - preloaded 7-6
  - static 7-6
- apply changes to a remote agent 10-26
- Apply Configuration Dialog Box 10-26

## B

- bind
  - IP Address 5-5
- BMC PATROL Agent Console COM Server
  - C-4
- BMC PATROL Agent Service COM Server
  - C-4

## C

- cache, event 11-4
- cfg file extension B-2
- Change Entry Dialog Box 10-17

- change file 3-4, 9-12
  - adding new variables 9-36
  - creating new 9-21
  - default account variable 9-49
  - deleting variables 9-38
  - format B-3
  - handling 9-18
  - opening 9-19
  - viewing 9-23
- character classes
  - examples F-5
- collected parameter values
  - days kept 12-8
- COM compatible applications C-1
- command line configuration utility 8-2
- Component Object Model (COM)
  - compatible applications C-1
- components
  - PATROL Cluster Configuration Wizard
    - 6-14
- comSecurityLevel C-3
- comUserAllowed C-3
- config.default 3-2, 9-3, 9-5, 10-4
  - reloading 9-15
- configuration
  - SNMP 14-5
    - SNMP changes go into affect 15-4
- Configuration File Dialog Box 9-20
- configuration files 3-4
  - backing up 5-9
- configuration variables A-1
- configuring
  - PATROL SNMP PATROL Master Agent
    - 15-4
- configuring the PATROL Agent
  - at the command line 8-2
    - examples of 8-10
    - prerequisites 8-2
    - syntax 8-3
  - for SNMP 14-8

- connect
  - reconnect in DHCP environment 5-8
- console
  - relationship managed by agent 5-2
  - start agent without a connection to 5-4
  - Unix display environment 5-3
- control
  - access to the agent 5-9
  - processes
    - number created by agent H-19

## D

- data points
  - kept in cache 12-7
- DCOM Configuration Utility C-4
- DCOM interface
  - configuring C-2
- DCOM programming interface C-1
- DCOMCNFG.EXE C-4
- debug
  - PSL H-21
- debugging, PSL for SNMP 16-9
- default account 13-2
- default configuration
  - reloading 3-5
- default values 3-5
- delay
  - for a process
    - maximum H-18
- deleting a variable from a configuration
  - 10-20
- DHCP
  - PATROL Agent conditions 5-8
  - PATROL Console conditions 5-8
  - PATROL support for 5-8
- Diag 15-16
- directories
  - \$PATROL\_HOME 1-10

- changing ownership and permissions 13-6
- disabled
  - applications 7-6
- disabled KMs 7-6, 7-10, 7-11
- Disconnect 15-16
- discovery rate
  - for applications H-9
- DISPLAY environment variable 5-3
- Distributed Component Object Model
  - programming interface C-1
- dump\_events. See utilities
- dump\_hist. See utilities
- dynamic
  - applications 7-6
- dynamic KMs 7-6

## E

- Edit Variable Dialog Box 9-40
  - deleting a change entry 9-48
  - new change entry 9-44
- Engine, PEM 1-14
- error
  - set PSL runtime error level H-21
- error log
  - for PATROL Agent 11-16
- error messages, PSL functions 16-10
- event app
  - UnregAllApp 15-17
- event cache 11-4
  - flushing 12-7
  - size of 11-6
- event class
  - Diag 15-16
  - EventArchive 15-16
  - R3FilterData 15-16
  - RegApp 15-16
  - RemProcess 15-16
  - RemPsl 15-16

- Unload 15-16
- UpdAppState 15-17
- UpdInstState 15-17
- UpdMachineState 15-17
- UpdParState 15-17
- WorstApp 15-17
- event ID
  - filtering for A-5
- event log
  - maximum size of 11-5
- EventArchive 15-16
- events
  - amount kept in memory 11-7
- Extended C-6

## F

- failover
  - behavior 6-7
  - tolerance by PATROL Agent 6-6
- file
  - script
    - example G-4
    - userrc.sh G-3
- filename metacharacters. See wildcards
- files
  - \$PATROL\_HOME/bin 1-12, 1-13
  - \$PATROL\_HOME/lib/commit.lock 1-12
  - \$PATROL\_HOME/lib/config.default 1-12
  - \$PATROL\_HOME/lib/knowledge 1-12
  - \$PATROL\_HOME/lib/knowledge/StdEv ents.ctg 1-12
  - \$PATROL\_HOME/lib/psl 1-12
  - \$PATROL\_HOME/log/history///dir.dat, dir.idx, param.hist 1-12
  - \$PATROL\_HOME/log/PatrolAgent--.err s~~ 1-13
  - \$PATROL\_HOME/log/PEM\_-.log, archive 1-12

\$PATROL\_HOME/patrol/config\_-.dat,  
idx 1-12  
/tmp/patrol/PEM\_-.log.lock 1-13  
PATROL Agent, list of 1-11

## filter

applications 7-19  
implicit 7-12  
precedence 7-13  
types used by PATROL Agent 7-12

filtered application instances variable 7-19

filtered regular expressions variable 7-19

## filtering

for event ID A-5

## firewall

port number 4-9

fix\_hist. See utilities

## Forests

managing rights 13-11

format for change files B-3

## G

get configuration 10-10

Get Configuration Dialog Box 10-10

## H

host list 9-33

add a host 9-33

## I

implicit filter 7-12

instance state changes 16-11

instance status 16-12

## instructions

maximum number for PSL process H-6

internal scheduling priority H-13

IP address

set for agent 5-5

## K

### KMs

disabled 7-6, 7-10, 7-11

by application name 7-10

by architecture 7-11

dynamic 7-6

loading status 7-6

preloaded 7-6, 7-7

by application name 7-7

by architecture 7-8

static 7-6, 7-9

### Knowledge Modules

disabled 7-6, 7-10, 7-11

by application name 7-10

by architecture 7-11

dynamic 7-6

loading status 7-6

preloaded 7-6, 7-7

by application name 7-7

by architecture 7-8

static 7-6, 7-9

## L

Library C-6

### limit

processes created by agent H-19

### loading

status

applications 7-6

KMs 7-6

### log

agent audit 11-21

PATROL Agent Error 11-16

PATROL Event 11-3

- PATROL Event Archive 11-15
- PEM
  - suppressing messages 5-2
- Single Sign-On 11-20

## M

- managers list
  - changing 14-11
- matching
  - pattern F-2
- messages
  - PEM
    - suppressing 5-2
- metacharacters. See wildcards
- MIB file 15-31, A-12
  - PATROL Agent objects D-1
  - scalar variables D-2
  - syntax types D-5
  - tables D-1
  - variables 16-3
- Microsoft DCOM Configuration Tool (DCOMCNFG.EXE) C-4
- Modify Variable Dialog Box 10-16, 10-19
- monitor
  - applications 7-18
    - include or exclude filters 7-18

## N

- network card
  - multiple 5-5
- nice value H-12, H-13
  - for PATROL Agent process H-12
  - for process that refreshes the process cache H-13
  - for processes created by the PATROL Agent H-12
- NO\_TRAP 15-15

- Non-PATROL components 1-9

## O

- objects D-1
- OnNow E-2
- operating system
  - scheduling priority H-12, H-13
- Operator Console 3-6

## P

- parameter last set time 16-12
- parameter status 16-12
- parameter value 16-12
- PATROL 1-2
  - 3.x architecture 1-3
  - 7.x architecture 1-4
  - components 1-6
    - DHCP support 5-8
- PATROL Agent 1-2, 1-8, C-1
  - application account 4-3
  - audit log 11-21
  - conditions for DHCP support 5-8
  - conditions under which a PATROL Agent will not monitor applications 7-3
  - configuration 3-1, 3-9
  - configuring for SNMP 15-9
  - configuring for Unix 9-3
  - connections with PATROL Console 5-4
  - default account 4-2
  - disabling support for SNMP traps 15-31
  - error log 11-16
  - failover behavior 6-7
  - failover tolerance 6-6
  - files 1-11
  - filter processing 7-12
  - instance account 4-4

- killing 9-17
- MIB 16-12
- monitoring applications
  - exclude 7-18
  - include 7-18
- nice value H-12
- OpenVMS
  - starting 2-27
  - stopping 2-30
- operating system scheduling priority H-12
- power management support E-4
- processes
  - limit number H-19
- processes created by
  - nice value H-12
  - operating system scheduling priority H-12
- purging configuration 3-4
- reinitializing 9-12
- relationship with console 5-2
- retrieving a configuration 10-8
- run queue
  - gap H-17
  - gap increment H-17
- security 13-2
- services 1-14
- SNMP role 14-4
- SNMP roles 14-8
- start without a console connection 5-4
- starting 2-17
- status
  - loading applications 7-6
  - loading KMs 7-6
- Unix 2-11, 9-1
- variables 3-10
- Windows 2-16, 2-24, 10-1
- PATROL Agent Configuration Utility 2-16
- PATROL Central Operator - Microsoft Windows Edition 1-6
- PATROL Central Operator - Web Edition 1-6
- PATROL Client Configuration Dialog Box
  - ESI Library Tab C-7
  - Type Library Tab C-7
- PATROL Client Configuration Utility (PCCFG.EXE) C-6
- PATROL Cluster Configuration Wizard
  - description 6-14
- PATROL Command Line Interface 1-9
- PATROL Console 9-14
  - conditions for DHCP support 5-8
  - reconnecting in DHCP environment 5-8
  - SNMP role 14-4
  - Unix display environment 5-3
- PATROL Console Server 1-6
- PATROL DCOM Registration Utility C-6
- PATROL Developer Console 1-7, 9-8
- PATROL Event Archive 11-15
- PATROL Event log 11-3
- PATROL Event Manager (PEM Console) 1-7, 15-15
- PATROL Events
  - method of sending SNMP traps 14-11
- PATROL KM 1-8, 3-6, 16-12
- PATROL Knowledge Module 1-8
- PATROL Master Agent
  - changing configuration 15-4
  - configuration file 15-6
  - configuration file directory 15-6
  - directory of executable file 15-5
  - executable name 15-5
  - nonvolatile information file 15-7
  - nonvolatile information file directory 15-7
  - port number 15-12
  - working directory 15-8
- PATROL MIB Tables 16-12
- PATROL Operator Console 1-7
- PATROL Script Language (PSL) 1-9

- method of sending SNMP traps 14-11
- PATROL SNMP Master Agent
  - configuring 15-4
  - read community string 15-11
  - start line 15-4
  - write community string 15-12
- PATROL SNMP Sub-agent
  - autostart sub-agent 15-9
  - send traps directly to manager 15-14
  - send traps through master agent 15-13
  - SNMP role 14-4
  - start and stop with PSL 14-10
- PATROL SNMP sub-agent
  - configuring 15-9
- patrol.conf, allowsendparamonly 5-21
- patrol.mib file D-1
- PATROL\_ADMIN environment variable 13-3
- PATROL\_MAX\_FILE\_DESCRIPTOR H-19
- PATROL\_PORT 4-8
- PATROLVIEW Products 1-8
- pattern matching F-2
- PCCFG.EXE C-6
- pconfig 3-10, 8-2
  - examples 8-10
  - prerequisites 8-2
  - syntax 8-3
- PEM 15-15
  - log
    - suppress messages 5-2
    - suppressing messages 5-2
- PEM Engine 1-16
  - services 1-17
- PEM Log 11-3
- pem.log file name 11-5
- port
  - firewall 4-9
  - local 4-9
- port number
  - indicating 2-11, 2-27, 2-30
- power management
  - ACPI support E-2
  - OnNow support E-2
  - PATROL Agent support E-4
  - sleep states E-3
- precedence
  - filter 7-13
- preloaded
  - applications 7-6
- preloaded KMs 7-6, 7-7
- primary window
  - handling variables 9-32
- privileges 2-17
- process
  - delay H-18
  - interactive 2-17
  - service 2-17
- process cache
  - nice value H-13
  - operating system scheduling priority H-13
  - refresh rate H-9
- processes
  - PATROL Agent
    - limit number created by H-19
- PSL 1-9, 14-11, 16-2
  - debug H-21
  - registered SNMP manager list 14-11
  - runtime error level H-21
  - SNMP debugging 16-9
  - SNMP error messages 16-10
  - starting and stopping the SNMP Agent 14-10
- PSL Compiler 1-9
- PSL Debugger 1-9
- PSL process
  - maximum number of instructions H-6
  - time period H-7
- PslSet

- event class
  - PslSet 15-16
- purge (delete) configuration, 9-28

## Q

- quantifiers
  - examples F-6

## R

- R3FilterData 15-16
- rate
  - discovery for applications H-9
  - refresh for process cache H-9
- refresh rate
  - for process cache H-9
- RegApp 15-16
- registered SNMP manager list 14-11
- regular expressions F-2
  - wildcards F-3
- relationship
  - agent to console 5-2
- RemProcess 15-16
- RemPsl 15-16
- resetting a variable to Its default value 9-40
- Result
  - event class
    - Result 15-16
- RTSERVERS environment variable 2-6, A-18
- run queue
  - gap H-17
  - gap increment H-17

## S

- scheduling policy

- for the PATROL Agent H-15
- scheduling priority
  - internal H-13
  - operating system H-12, H-13
- script file
  - userrc.sh
    - example G-4
- security
  - Access Control List 13-2
  - default account 13-2
  - security considerations 2-17
  - Security levels 13-2
  - SEND\_TRAP 15-15
  - sending new license 9-30
  - sending variables 9-22
  - Set Default Account Dialog Box 10-19
  - setting MIB variables 16-3
  - sleep states E-3
  - snm\_h\_set() 16-8
- SNMP
  - communication 14-8
  - dependencies 14-4
  - listening for traps with PSL 16-2
  - manager list
    - register with PSL 14-11
  - manipulating MIB with PSL 16-3
  - Master Agent working directory on
    - Unix 15-8
  - MIB file A-12
  - MIB II information 15-13
  - minimal configuration 14-5
  - PATROL architecture 14-3
- PEM
  - end time 15-26
  - filter events by class 15-24
  - filter events by description 15-25
  - node name 15-23
  - severity level 15-22
  - start time 15-26
  - status mask 15-28

- support based on PEM 15-21
- trap format 15-29
- type mask 15-27
- PSL built-in functions 16-8
- roles of PATROL Agent 14-8
- starting Master Agent on Unix 15-4
  - configuration file directory 15-6
  - configuration file name 15-6
  - directory name 15-5
  - executable name 15-5
  - nonvolatile information file 15-7
  - nonvolatile information file
    - directory 15-7
- starting PATROL SNMP Master Agent 15-8
- startup process 14-6
- support 1-9
- SNMP get operations 16-12
- SNMP support
  - components 14-4
- SNMP Traps
  - destination 15-20
  - list of recipients 15-20
- SNMP traps
  - directly to SNMP Manager 15-14
  - methods of sending 14-11
    - comparison 14-12
  - through PATROL SNMP Master Agent 15-13
- snmp\_agent\_config() 16-8
- snmp\_agent\_start() 16-8
- snmp\_agent\_stop() 16-8
- snmp\_close() 16-8
- snmp\_config() 16-8
- snmp\_get() 16-8
- snmp\_get\_next() 16-8
- snmp\_h\_get\_next() 16-8
- snmp\_h\_set() 16-8
- snmp\_open() 16-8
- snmp\_raise\_std\_trap() 16-8

- snmp\_set() 16-8
- snmp\_trap\_ignore() 16-8
- snmp\_trap\_listen() 16-8
- snmp\_trap\_receive() 16-8
- snmp\_trap\_register\_im() 16-8
- snmp\_trap\_send() 15-20, 16-9
- snmp\_walk() 16-9
- SNMPStart 14-7, 15-8
- standard event classes 15-16
- starting the PATROL Agent
  - SNMP Agent 14-10
- static
  - applications 7-6
- static KMs 7-6
- status
  - application 7-6
  - KM 7-6
  - Knowledge Modules 7-6
- stopping the PATROL Agent
  - SNMP Agent 14-10

## T

- tables, MIB file
  - applicationsTable D-11
  - applInstTable D-14
  - objectsTable D-6
  - parametersTable D-18
  - piV1mTable D-24
  - variablesTable D-8
- TCP/IP port number
  - indicating 2-11, 2-27, 2-30
- time period
  - for PSL process H-7
- trap listening
  - PSL 16-2
- trap sending 16-9
  - PSL 14-10
- TRAP\_SEND 14-11
- traps, Agent types D-3

## U

- Unix
  - display environment 5-3
  - environment variables
    - storage G-2
- unix output 5-3
- Unload 15-16
- UnregAllApp 15-17
- UpdAppState 15-17
- Update Connection 9-14
  - in DHCP environment 5-8
- UpdInstState 15-17
- UpdMachineState 15-17
- UpdParState 15-17
- user-defined variables 3-12
- userrc.sh
  - format G-3
  - example G-4
- utilities 12-1–12-19
  - dump\_events 11-8
    - directories 11-9
    - example
      - dump\_events 11-14
      - limitations and guidelines 11-9
      - options 11-12
      - syntax 11-11
      - troubleshooting 11-14
  - dump\_hist 11-8, 12-10
    - directories 12-11, 12-22
    - examples 12-18, 12-27
    - options 12-15, 12-25
    - syntax 12-14, 12-25
  - fix\_hist 12-20
    - examples 12-27
    - options 12-25
    - recovery actions 12-24
    - syntax 12-25

## V

- variable
  - /AgentSetup/AgentTuning/runqDeltaIncrement H-17
  - environment
    - DISPLAY 5-3
    - RTSERVERS 2-6, A-18
- variables
  - /AgentSetup/.OSdefaultAccount 4-4
  - /AgentSetup/.OSdefaultAccount 4-3
  - /AgentSetup/.OSdefaultAccountAppliesToCmds 4-5
  - /AgentSetup/\_name\_ 3-7
  - /AgentSetup/\_type\_ 3-7
  - /AgentSetup/accessControlList 5-9
  - /AgentSetup/AgentTuning/\_name 3-7
  - /AgentSetup/AgentTuning/\_type\_ 3-7
  - /AgentSetup/AgentTuning/agentPriority H-12
  - /AgentSetup/AgentTuning/applCheckCycle H-9
  - /AgentSetup/AgentTuning/getProcsCycle H-9
  - /AgentSetup/AgentTuning/procCachePriority H-13
  - /AgentSetup/AgentTuning/procCacheScheduledPriority H-13
  - /AgentSetup/AgentTuning/pslInstructionMax H-6
  - /AgentSetup/AgentTuning/pslInstructionPeriod H-7
  - /AgentSetup/AgentTuning/runqDelta H-17
  - /AgentSetup/AgentTuning/runqMaxDelta H-18
  - /AgentSetup/AgentTuning/runqSchedPolicy H-15
  - /AgentSetup/AgentTuning/userPriority H-12

/AgentSetup/auditLog 11-23  
 /AgentSetup/BindToAddress 5-5  
 /AgentSetup/defaultAccount 4-2, 10-18  
 /AgentSetup/defaultAccountShell 4-2  
 /AgentSetup/defaultDisplay 5-3  
 /AgentSetup/disabledKMs 7-10  
 /AgentSetup/disabledKMsArch 7-11  
 /AgentSetup/EnableSysOutputAclCheck 5-13  
 /AgentSetup/historyRetentionPeriod 12-8  
 /AgentSetup/localPortForRemoteOpen 4-9  
 /AgentSetup/maxAgentMessageLimit 11-18  
 /AgentSetup/maxProcessLimit H-20  
 /AgentSetup/pemCacheSize 11-6  
 /AgentSetup/pemEvMemRetention 11-7  
 /AgentSetup/pemLogName 11-5, 12-7  
 /AgentSetup/pemSnmSupport 15-21  
 /AgentSetup/PerfGetDebugFile H-23  
 /AgentSetup/PerfGetMaxTimes H-23  
 /AgentSetup/PerfGetTimeout H-22  
 /AgentSetup/PerfMaxRestartCount H-20  
 /AgentSetup/PortConnectType 5-6, 5-7  
 /AgentSetup/preloadedKMs 7-7  
 /AgentSetup/preloadedKMsArch 7-8  
 /AgentSetup/prmHistCache FlushTimer 12-7  
 /AgentSetup/prmHistCacheMaxFlushTime 12-8  
 /AgentSetup/prmHistCacheSize 12-7  
 /AgentSetup/PsIdDebug H-21  
 /AgentSetup/sessionsInitMayFail 5-4  
 /AgentSetup/staticApplications 7-9  
 /AgentSetup/suppressConsoleInfoMsgMask 5-2  
 /AgentSetup/suppressSystemOutputMsgMask 5-3  
 /AgentSetup/timeZone 4-4  
 /AgentSetup/trustedConnectionsAccount 4-7  
 /AgentSetup/XPC/.xpc\_defaultAccount 4-5  
 /numLicenseDays 3-12  
 /snmp/\_name\_ 3-7  
 /snmp/\_type\_ 3-7  
 /snmp/agent\_auto\_start 15-9, 15-11  
 /snmp/agent\_r\_community 15-11  
 /snmp/agent\_w\_community 15-12  
 /snmp/default\_port 16-7  
 /snmp/default\_r\_community 16-5  
 /snmp/default\_retries 16-7  
 /snmp/default\_timeout 16-6  
 /snmp/default\_w\_community 16-6  
 /snmp/ignoreIPAddress 15-14  
 /snmp/masteragent\_auto\_start 15-11  
 /snmp/masterAgentConfigDir 15-6  
 /snmp/masterAgentConfigName 15-6  
 /snmp/masterAgentDir 15-5  
 /snmp/masterAgentName 15-5  
 /snmp/masterAgentParamDir 15-7  
 /snmp/masterAgentParamName 15-7  
 /snmp/masterAgentStartLine 15-4  
 /snmp/masterAgentWorkingDir 15-8  
 /snmp/piVIm\_list 15-20  
 /snmp/trap\_port 15-12  
 /snmp/trapConfTable 15-13  
 /snmp/trapMibTable 15-14  
 agentExecuteCommand D-2  
 AgentSetup/pemLogSize 11-5  
 allowsendparamonly 5-21  
 changing  
     determining the method for 3-11  
 creating user-defined 3-12  
 environment  
     storing in Unix script file G-2  
 MIB 16-3  
 objectsCwd D-2

- PATROL\_MAX\_FILE\_DESCRIPTOR  
H-19
- PATROL\_PORT 4-8
- SNMP support 16-3
- variables, Agent setup
  - filtered application instances 7-19
  - filtered regular expressions 7-19
- version arbitration 7-4
  - and KMs 7-4
  - when a PATROL Developer Console connects to an Agent 7-5
  - when a PATROL Operator Console connects to an Agent 7-4
- View Changes dialog box 10-24

## W

- wildcards F-3, F-4
  - \$ F-4
  - () F-4
  - \* F-4
  - + F-4
  - . F-4
  - < F-4
  - > F-4
  - ? F-4
  - ^ F-4
- anchors
  - examples F-7
- character classes
  - examples F-5
- quantifiers
  - examples F-6
- WorstApp 15-17
- wpconfig 3-10, 10-2, C-3
  - add new variable 10-11
  - apply changes 10-26
  - delete variable 10-21
  - functions 10-3
  - modify variable 10-15

- modifying the default account variable 10-18
- new configuration 10-8
- tasks 10-8
- view changes 10-22
- wpconfig Primary Window 10-6
- wpconfig Window
  - closing 10-8

## X

- xpconfig 3-10, 9-3
  - apply configuration 9-12
  - command line 9-9
  - functions 9-5
  - prerequisites 9-4
  - primary window 9-7
  - syntax 9-9
  - variables box 9-7

# END USER LICENSE AGREEMENT NOTICE

**BY OPENING THE PACKAGE, INSTALLING, PRESSING "AGREE" OR "YES" OR USING THE PRODUCT, THE ENTITY OR INDIVIDUAL ENTERING INTO THIS AGREEMENT AGREES TO BE BOUND BY THE FOLLOWING TERMS. IF YOU DO NOT AGREE WITH ANY OF THESE TERMS, DO NOT INSTALL OR USE THE PRODUCT, PROMPTLY RETURN THE PRODUCT TO BMC OR YOUR BMC RESELLER, AND IF YOU ACQUIRED THE LICENSE WITHIN 30 DAYS OF THE DATE OF YOUR ORDER CONTACT BMC OR YOUR BMC RESELLER FOR A REFUND OF LICENSE FEES PAID. IF YOU REJECT THIS AGREEMENT, YOU WILL NOT ACQUIRE ANY LICENSE TO USE THE PRODUCT.**

This Agreement ("**Agreement**") is between the entity or individual entering into this Agreement ("You") and BMC Software Distribution, Inc., a Delaware corporation located at 2101 CityWest Blvd., Houston, Texas, 77042, USA or its affiliated local licensing entity ("BMC"). "You" includes you and your Affiliates. "Affiliate" is defined as an entity which controls, is controlled by or shares common control with a party. THIS AGREEMENT WILL APPLY TO THE PRODUCT, UNLESS (1) YOU AGREED TO A WEB BASED LICENSE AGREEMENT WITH BMC WHEN ORDERING THE PRODUCT, IN WHICH CASE THAT WEB BASED LICENSE AGREEMENT GOVERNS THE USE OF THE PRODUCT, OR (2) IF YOU DID NOT AGREE TO A WEB BASED LICENSE AGREEMENT WITH BMC WHEN ORDERING THE PRODUCT AND YOU HAVE A WRITTEN LICENSE AGREEMENT WITH BMC, THEN THAT WRITTEN AGREEMENT GOVERNS THE USE OF THE PRODUCT. THE ELECTRONIC AGREEMENT PROVIDED WITH THE PRODUCT AS PART OF THE INSTALLATION OF THE PRODUCT WILL NOT APPLY. In addition to the restrictions imposed under this Agreement, any other usage restrictions contained in the Product installation instructions or release notes shall apply to Your use of the Product.

**PRODUCT AND CAPACITY. "Software"** means the object code version of the computer programs provided, via delivery or electronic transmission, to You. Software includes computer files, enhancements, maintenance modifications, upgrades, updates, bug fixes, and error corrections.

**"Documentation"** means all written or graphical material provided by BMC in any medium, including any technical specifications, relating to the functionality or operation of the Software.

**"Product"** means the Software and Documentation.

**"License Capacity"** means the licensed capacity for the Software with the pricing and other license defining terms, including capacity restrictions, such as tier limit, total allowed users, gigabyte limit, quantity of Software, and/or other capacity limitations regarding the Software. For licenses based on the power of a computer, You agree to use BMC's current computer classification scheme, which is available at <http://www.bmc.com> or can be provided to You upon request.

**ACCEPTANCE.** The Product is deemed accepted by You, on the date that You received the Product from BMC.

**LICENSE.** Subject to the terms of this Agreement, as well as Your payment of applicable fees, BMC grants You a non-exclusive, non-transferable, perpetual (unless a term license is provided on an order) license for each copy of the Software, up to the License Capacity, to do the following:

- (a) install the Software on Your owned or leased hardware located at a facility owned or controlled by You in the country where You acquired the license;
- (b) operate the Software solely for processing Your own data in Your business operations; and
- (c) make one copy of the Software for backup and archival purposes only (collectively a "**License**").

If the Software is designed by BMC to permit you to modify such Software, then you agree to only use such modifications or new software programs for Your internal purposes or otherwise consistent with the License. BMC grants You a license to use the Documentation solely for Your internal use in Your operations.

**LICENSE UPGRADES.** You may expand the scope of the License Capacity only pursuant to a separate agreement with BMC for such expanded usage and Your payment of applicable fees. There is no additional warranty period or free support period for license upgrades.

**RESTRICTIONS:** You agree to **NOT**:

- (a) disassemble, reverse engineer, decompile or otherwise attempt to derive any Software from executable code;
- (b) distribute or provide the Software to any third party (including without limitation, use in a service bureau, outsourcing environment, or processing the data of third parties, or for rental, lease, or sublicense); or
- (c) provide a third party with the results of any functional evaluation or benchmarking or performance tests, without BMC's prior written approval, unless prohibited by local law.

**TRIAL LICENSE.** If, as part of the ordering process, the Product is provided on a trial basis, then these terms apply: (i) this license consists solely of a non-exclusive, non-transferable evaluation license to operate the Software for the period of time specified from BMC or, if not specified, a 30 day time period ("**Trial Period**") only for evaluating whether You desire to acquire a capacity-based license to the Product for a fee; and (ii) Your use of the Product is on an AS IS basis without any warranty, and **BMC, ITS AFFILIATES AND RESELLERS, AND LICENSORS DISCLAIM ANY AND ALL WARRANTIES (INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT) AND HAVE NO LIABILITY WHATSOEVER RESULTING FROM THE USE OF THIS PRODUCT UNDER THIS TRIAL LICENSE ("Trial License").** BMC may terminate for its convenience a Trial License upon notice to You. When the Trial Period ends, Your right to use this Product automatically expires. If You want to continue Your use of the Product beyond the Trial Period, contact BMC to acquire a capacity-based license to the Product for a fee.

**TERMINATION.** This Agreement shall immediately terminate if You breach any of its terms. Upon termination, for any reason, You must uninstall the Software, and either certify the destruction of the Product or return it to BMC.

**OWNERSHIP OF THE PRODUCT.** BMC or its Affiliates or licensors retain all right, title and interest to and in the BMC Product and all intellectual property, informational, industrial property and proprietary rights therein. BMC neither grants nor otherwise transfers any rights of ownership in the BMC Product to You. BMC Products are protected by applicable copyright, trade secret, and industrial and intellectual property laws. BMC reserves any rights not expressly granted to You herein.

**CONFIDENTIAL AND PROPRIETARY INFORMATION.** The BMC Products are and contain valuable confidential information of BMC ("**Confidential Information**"). Confidential Information means non-public technical and non-technical information relating to the BMC Products and Support, including, without limitation, trade secret and proprietary information, and the structure and organization of the Software. You may not disclose the Confidential Information to third parties. You agree to use all reasonable efforts to prevent the unauthorized use, copying, publication or dissemination of the Product.

**WARRANTY.** Except for a Trial License, BMC warrants that the Software will perform in substantial accordance with the Documentation for a period of one year from the date of the order. This warranty shall not apply to any problems caused by software or hardware not supplied by BMC or to any misuse of the Software.

**EXCLUSIVE REMEDY.** BMC's entire liability, and Your exclusive remedy, for any defect in the Software during the warranty period or breach of the warranty above shall be limited to the following: BMC shall use reasonable efforts to remedy defects covered by the warranty or replace the defective Software within a reasonable period of time, or if BMC cannot remedy or replace such defective copy of the Software, then BMC shall refund the amount paid by You for the License for that Software. BMC's obligations in this section are conditioned upon Your providing BMC prompt access to the affected Software and full cooperation in resolving the claim.

**DISCLAIMER. EXCEPT FOR THE EXPRESS WARRANTIES ABOVE, THE PRODUCT IS PROVIDED "AS IS." BMC, ITS AFFILIATES AND LICENSORS SPECIFICALLY DISCLAIM ALL OTHER WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT. BMC DOES NOT WARRANT THAT THE OPERATION OF THE SOFTWARE WILL BE UNINTERRUPTED OR ERROR FREE, OR THAT ALL DEFECTS CAN BE CORRECTED.**

**DISCLAIMER OF DAMAGES. IN NO EVENT IS BMC, ITS AFFILIATES OR LICENSORS LIABLE FOR ANY SPECIAL, INDIRECT, INCIDENTAL, PUNITIVE OR CONSEQUENTIAL DAMAGES RELATING TO OR ARISING OUT OF THIS AGREEMENT, SUPPORT, AND/OR THE PRODUCT (INCLUDING, WITHOUT LIMITATION, LOST PROFITS, LOST COMPUTER USAGE TIME, AND DAMAGE OR LOSS OF USE OF DATA), EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, AND IRRESPECTIVE OF ANY NEGLIGENCE OF BMC OR WHETHER SUCH DAMAGES RESULT FROM A CLAIM ARISING UNDER TORT OR CONTRACT LAW.**

**LIMITS ON LIABILITY. BMC'S AGGREGATE LIABILITY FOR DAMAGES IS LIMITED TO THE AMOUNT PAID BY YOU FOR THE LICENSE TO THE PRODUCT.**

**SUPPORT.** If Your order includes support for the Software, then BMC agrees to provide support (24 hours a day/7 days a week) ("**Support**"). You will be automatically re-enrolled in Support on an annual basis unless BMC receives notice of termination from You as provided below. There is a free support period during the one year warranty period.

(a) **Support Terms.** BMC agrees to make commercially reasonable efforts to provide the following Support: (i) For malfunctions of supported versions of the Software, BMC provides bug fixes, patches or workarounds in order to cause that copy of the Software to operate in substantial conformity with its then-current operating specifications; and (ii) BMC provides new releases or versions, so long as such new releases or versions are furnished by BMC to all other enrolled Support customers without additional charge. BMC may refuse to provide Support for any versions or releases of the Software other than the most recent version or release of such Software made available by BMC. Either party may terminate Your enrollment in Support upon providing notice to the other at least 30 days prior to the next applicable Support anniversary date. If You re-enroll in Support, BMC may charge You a reinstatement fee of 1.5 times what You would have paid if You were enrolled in Support during that time period.

(b) **Fees.** The annual fee for Support is 20% of the Software's list price less the applicable discount or a flat capacity based annual fee. BMC may change its prices for the Software and/or Support upon at least 30 days notice prior to Your support anniversary date.

**VERIFICATION.** If requested by BMC, You agree to deliver to BMC periodic written reports, whether generated manually or electronically, detailing Your use of the Software in accordance with this Agreement, including, without limitation, the License Capacity. BMC may, at its expense, audit Your use of the Software to confirm Your compliance with the Agreement. If an audit reveals that You have underpaid fees, You agree to pay such underpaid fees. If the underpaid fees exceed 5% of the fees paid, then You agree to also pay BMC's reasonable costs of conducting the audit.

**EXPORT CONTROLS.** You agree not to import, export, re-export, or transfer, directly or indirectly, any part of the Product or any underlying information or technology except in full compliance with all United States, foreign and other applicable laws and regulations.

**GOVERNING LAW.** This Agreement is governed by the substantive laws in force, without regard to conflict of laws principles: (a) in the State of New York, if you acquired the License in the United States, Puerto Rico, or any country in Central or South America; (b) in the Province of Ontario, if you acquired the License in Canada (subsections (a) and (b) collectively referred to as the "**Americas Region**"); (c) in Singapore, if you acquired the License in Japan, South Korea, Peoples Republic of China, Special Administrative Region of Hong Kong, Republic of China, Philippines, Indonesia, Malaysia, Singapore, India, Australia, New Zealand, or Thailand (collectively, "**Asia Pacific Region**"); or (d) in the Netherlands, if you acquired the License in any other country not described above. The United Nations Convention on Contracts for the International Sale of Goods is specifically disclaimed in its entirety.

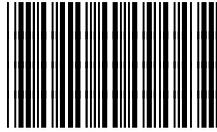
**ARBITRATION. ANY DISPUTE BETWEEN YOU AND BMC ARISING OUT OF THIS AGREEMENT OR THE BREACH OR ALLEGED BREACH, SHALL BE DETERMINED BY BINDING ARBITRATION CONDUCTED IN ENGLISH. IF THE DISPUTE IS INITIATED IN THE AMERICAS REGION, THE ARBITRATION SHALL BE HELD IN NEW YORK, U.S.A., UNDER THE CURRENT COMMERCIAL OR INTERNATIONAL, AS APPLICABLE, RULES OF THE AMERICAN ARBITRATION ASSOCIATION. IF THE DISPUTE IS INITIATED IN A COUNTRY IN THE ASIA PACIFIC REGION, THE ARBITRATION SHALL BE HELD IN SINGAPORE, SINGAPORE UNDER THE CURRENT UNCITRAL ARBITRATION RULES. IF THE DISPUTE IS INITIATED IN A COUNTRY OUTSIDE OF THE AMERICAS REGION OR ASIA PACIFIC REGION, THE ARBITRATION SHALL BE HELD IN AMSTERDAM, NETHERLANDS UNDER THE CURRENT UNCITRAL ARBITRATION RULES. THE COSTS OF THE ARBITRATION SHALL BE BORNE EQUALLY PENDING THE ARBITRATOR'S AWARD. THE AWARD RENDERED SHALL BE FINAL AND BINDING UPON THE PARTIES AND SHALL NOT BE SUBJECT TO APPEAL TO ANY COURT, AND MAY BE ENFORCED IN ANY COURT OF COMPETENT JURISDICTION. NOTHING IN THIS AGREEMENT SHALL BE DEEMED AS PREVENTING EITHER PARTY FROM SEEKING INJUNCTIVE RELIEF FROM ANY COURT HAVING JURISDICTION OVER THE PARTIES AND THE SUBJECT MATTER OF THE DISPUTE AS NECESSARY TO PROTECT EITHER PARTY'S CONFIDENTIAL INFORMATION, OWNERSHIP, OR ANY OTHER PROPRIETARY RIGHTS. ALL ARBITRATION PROCEEDINGS SHALL BE CONDUCTED IN CONFIDENCE, AND THE PARTY PREVAILING IN ARBITRATION SHALL BE ENTITLED TO RECOVER ITS REASONABLE ATTORNEYS' FEES AND NECESSARY COSTS INCURRED RELATED THERETO FROM THE OTHER PARTY.**

**U.S. GOVERNMENT RESTRICTED RIGHTS.** The Software under this Agreement is "commercial computer software" as that term is described in 48 C.F.R. 252.227-7014(a)(1). If acquired by or on behalf of a civilian agency, the U.S. Government acquires this commercial computer software and/or commercial computer software documentation subject to the terms of this Agreement as specified in 48 C.F.R. 12.212 (Computer Software) and 12.211 (Technical Data) of the Federal Acquisition Regulations ("**FAR**") and its successors. If acquired by or on behalf of any agency within the Department of Defense ("**DOD**"), the U.S. Government acquires this commercial computer software and/or commercial computer software documentation subject to the terms of this Agreement as specified in 48 C.F.R. 227.7202 of the DOD FAR Supplement and its successors.

**MISCELLANEOUS TERMS.** You agree to pay BMC all amounts owed no later than 30 days from the date of the applicable invoice, unless otherwise provided on the order for the License to the Products. You will pay, or reimburse BMC, for taxes of any kind, including sales, use, duty, tariffs, customs, withholding, property, value-added (VAT), and other similar federal, state or local taxes (other than taxes based on BMC's net income) imposed in connection with the Product and/or the Support. This Agreement constitutes the entire agreement between You and BMC and supersedes any prior or contemporaneous negotiations or agreements, whether oral, written or displayed electronically, concerning the Product and related subject matter. No modification or waiver of any provision hereof will be effective unless made in a writing signed by both BMC and You. You may not assign or transfer this Agreement or a License to a third party without BMC's prior written consent. Should any provision of this Agreement be invalid or unenforceable, the remainder of the provisions will remain in effect. The parties have agreed that this Agreement and the documents related thereto be drawn up in the English language. Les parties exigent que la présente convention ainsi que les documents qui s'y rattachent soient rédigés en anglais.



# Notes



23692