

Installation and User Guide



Perl Module for PATROL®



Arackal Digital Solutions Inc.
240 Duncan Mill Rd., Suite 301
Toronto, ON, Canada
M3B 1Z4
T: 416.703.1211
T: 1.877.437.4933
F: 416.703.2544
info@arackal.com

Perl Module for PATROL®

Installation and User Guide

Covering Module Version 1.3.1

Aug 20, 2004

Document revision 1.04

This document is published by Arackal Digital Solutions
Copyright © 2002- 2004 Arackal Digital Solutions
All rights reserved.



Arackal Digital Solutions

Arackal Digital Solutions Inc.
240 Duncan Mill Rd., Suite 301
Toronto, ON, Canada
M3B 1Z4

BMC Software, the BMC Software logos and all other BMC Software product or service names are registered trademarks or trademarks of BMC Software, Inc. in the USA and in other select countries.

Contacting Arackal Digital Solutions

You can contact our support department via e-mail, fax or phone.

Website: <http://www.arackal.com> Telephone: (416)703.1211 or 1.877.437.4933 Fax: (416).703.2544

Contents

CONTENTS	III
ABOUT THIS GUIDE	1
WHO SHOULD READ THIS GUIDE	1
HOW THIS GUIDE IS STRUCTURED	1
MARGIN NOTE ICONS	1
RELATED PUBLICATIONS	2
CHAPTER 1: INTRODUCTION	3
PERL	3
WHY A PERL MODULE FOR PATROL?	3
HOW CAN YOU USE THE PERL MODULE FOR PATROL?	4
CHAPTER 2: GETTING STARTED	5
SYSTEM AND SOFTWARE VERIFICATION	5
DISK REQUIREMENTS	6
LICENSING	7
INSTALLATION AND CONFIGURATION	7
<i>Windows NT</i>	7
<i>UNIX</i>	7
TESTING THE INSTALLATION	8
TESTING WITH A PATROL AGENT	8
CHAPTER 3: USING THE PATPERL MODULE	11
LOADING THE MODULE	11
CONNECTING TO A PATROL AGENT	11
SAMPLE SCRIPT #1: GETTING THE CURRENT VALUE OF A PARAMETER	13
SAMPLE SCRIPT #2: GETTING A LIST OF LOADED KNOWLEDGE MODULES	14
SAMPLE SCRIPT #3: RUNNING A PSL SCRIPT REMOTELY	16
GETTING MORE INFORMATION	18

About This Guide

This guide provides details on the installation, configuration, and use of the Perl Module for PATROL®, hereafter referred to as “PatPerl”. It outlines the functionality provided by the product, as well as providing complete sample programs with explanations. This guide also provides details on performing the installation on the supported platforms.

It is useful to note that any screen shots present in this guide were taken from Windows NT. There should be no functional differences between these screens and those found in the UNIX version.

Who Should Read This Guide





This guide is intended for use by system and application administrators who are responsible for installing and configuring PATROL and PatPerl. It is somewhat technical in nature and assumes a fundamental knowledge of the PATROL architecture as well as familiarity with knowledge modules, PATROL Agent configuration, and most importantly, Perl5.

How This Guide Is Structured

Chapter	Title	Contents
1	Introduction	Introduces the Module
2	Getting Started	Provides installation and configuration information
3	Using the Module	How to use PatPerl to get work done
Appendix A		

Margin Note Icons

This manual makes use of notes in the left margin for presenting information that is useful or pertinent to the current discussion. The icons associated with these notes have the following meanings:

-  **Reference** A cross-reference to material found elsewhere in this manual.
-  **Example** An example of what was most recently discussed.
-  **Note** A note, or information of general interest.
-  **Warning** Warns that caution should be exercised when performing the associated actions.
- New** Indicates new or enhanced functionality that was not present in the previous release

Related Publications

All of the PATROL Installation Manuals, Release Notes, and Knowledge Module Guides are related to, and may be referenced in this guide. The NT or UNIX Console User Guide from BMC should be used as a reference for detailed information relating to the loading of Knowledge Modules, the setting of agent accounts, and items such as configuring the Agent access control list.

Any suggestions made within this guide are based upon the expertise of the author. The Perl scripts provided in this document are not guaranteed to be efficient or of “Perl-guru” level sophistication, but will work given a proper Active Perl5 installation.

Any suggestions or comments regarding this manual or the product should be directed to support@arackal.com.

Chapter 1: Introduction

Monitoring your environment using BMC Software PATROL® makes a lot of sense, even if you already have monitoring scripts of various origins sprinkled throughout your environment. PATROL brings many common monitoring requirements to the table:

- Automatic history gathering and storage
- Centralized, monitoring console
- Cross-platform, cron like scheduling
- A threshold, alarm, and event infrastructure

You can use PATROL as the infrastructure for running custom scripts, no matter what your scripting language may be. For many reasons, a large number of sites have chosen Perl as their system administration and monitoring scripting language.

Perl

One of the most widely accepted system administration languages in use today is Perl, the Practical Extraction and Report Language, written by Larry Wall. Thousands of system administrators, researchers, and mathematicians use Perl daily to perform a huge variety of tasks. Many great reasons to use Perl come from the Perl Conference 3.0 (1999):

“There are many reasons why Perl is a good choice, some of them are:

- Source code readily available
- no cost
- ported to just about every platform
- Perl has a wealth of documentation available
- commercial support *is* available
- Perl is great for getting jobs done quickly
- Perl is an excellent "glue" language that can be used to bring together many other programs, even if they are not written in Perl.

The bottom line here though is that Perl is a full-featured, commercially supported programming language that can reduce development time while providing excellent end-results.”

Perl is the ideal language for “gluing” together the many different systems present in a typical Enterprise Systems Management strategy.

Why a Perl Module for PATROL?

By bringing the world of Perl to PATROL, Arackal Digital Solutions joins the world’s most popular system administration language with the market leader in application and system management solutions. This marriage makes possible simple, elegant solutions for monitoring and reporting across your environment.

How can you use the Perl Module for PATROL?

The Perl Module for PATROL (hereafter referred to as “PatPerl”) was developed to bring the power of Perl to PATROL.

This section outlines several questions asked by Perl programmers upon first setting eyes on PATROL and how PatPerl can help provide simple answers.

“I already have Perl scripts monitoring my custom applications – how can I convert over to get PATROL to monitor these for me?”

Given the current PATROL architecture, there are two answers:

1. Setup a new parameter and have PATROL execute your Perl script for you, with any final value returned by your script being stored in the history database.

In many cases, returning a single value from the Perl script is acceptable – the number of lines found in a application log file, for instance. In more instances than not though, there are a number of metrics that a single script reports on, in which case this method is not really an option. You could employ a divide and conquer approach and split your single script up, running each of them as a separate parameter script. Ideally it would be great if the amount of script hacking could be kept to a minimum, so this solution isn’t great.

2. Setup a new parameter, convert your script to PATROL’s native Scripting Language (PSL), and toss your Perl script.

Most Perl programmers will find that PSL is fairly easy to learn – for simple scripts, removing the “\$” sign from in front of variable names will usually work. There are, however, many features and functions that Perl programmers have come to expect are not implemented in PSL – most notably Perl regular expressions, arrays (normal and associative), networking functions, and the hundreds of general purpose Perl Modules currently available. For many Perl scripts, converting to PSL would be hard, and in some cases impossible.

With PatPerl, your Perl scripts have complete access to the PATROL Agent’s namespace. This makes it simple to set parameter values, get PATROL variable values, send events, and even change the state of application icons – your Perl scripts are completely integrated into the PATROL environment.

“I’m an avid Perl programmer and I do a lot of CGI/web programming with it. How can I use Perl with PATROL to do useful things like generate status reports?”

Perl is the number one general reporting language in use today. There are Perl Modules freely available that allow you to easily interface with popular web servers, generate images, interact with SQL databases, and helpdesk software – this makes generating, displaying, and storing reports of all types fast, and simple.

PatPerl brings Perl’s reporting ability to PATROL. From a single host having PatPerl installed, it is possible to connect to many remote PATROL Agents and retrieve any number of statistics: parameter and application statuses, current values, KM versions, parameter histories, etc.

Chapter 2: Getting Started

This chapter provides a series of steps that should quickly get you started using the PatPerl module. Hardware and software pre-requisites, installation, and configuration are all covered.

System and Software Verification

The PatPerl module can be installed onto several different architectures, operating systems, and Perl5 versions, as listed in Table 1. For information on getting the latest version of Perl for your platform, check the Internet

Platform	Operating System	Perl Version
Intel x86	Win NT4/2000	Perl 5.6.1 Perl 5.8.1 Perl 5.8.3 ActivePerl 5.6.1 ActivePerl 5.8.1 ActivePerl 5.8.3
SPARC	Solaris2.7	ActivePerl 5.6.1 ActivePerl 5.8.1 ActivePerl 5.8.3
SPARC	Solaris2.8	Perl 5.6.1 Perl 5.8.1 Perl 5.8.3 ActivePerl 5.6.1 ActivePerl 5.8.1 ActivePerl 5.8.3
SPARC	Solaris2.9	Perl 5.6.1 Perl 5.8.1 Perl 5.8.3 ActivePerl 5.6.1 ActivePerl 5.8.1 ActivePerl 5.8.3
HPUX	11.11	Perl 5.6.1 Perl 5.8.1 Perl 5.8.3
AIX	4.3	Perl 5.6.1 Perl 5.8.1 Perl 5.8.3

Table 1: Supported Perl5 versions and Hardware Platforms

PatPerl will interface with PATROL Agents v3.4 and higher, running on standard UNIX and Windows NT/2000 based hosts.

Disk Requirements

Installing the module will take up space on your hard disk as outlined below.

Platform	Operating System	Disk Space
Intel x86	Win NT4/2000	2.5M
SPARC	Solaris	1.5M
HPUX	11.11	750k
AIX	4.3	1.5M

Table 2: Disk space requirements

The Perl Module does not require PATROL to be installed on the local machine.

Licensing

The PatPerl module is a licensed product. Thirty (30) day demo licenses are available at no charge from Arackal Digital Solutions via: support@arackal.com or automatically over the web by registering at <http://www.arackal.com>.

The installation of the license is covered in the installation section that follows.

Installation and Configuration

Installing the PatPerl module essentially involves copying the distribution files to the appropriate places in your Perl installation hierarchy and installing the license. UNIX and WinNT installations vary slightly and are outlined below.

Windows NT

1. Ensure that ActivePerl 5.6.1 or later (provided by ActiveState Corp) has been installed onto your machine. The path should be properly setup so you can execute "perl -v" from your MSDOS command prompt while in any directory and see the Perl5 version and Copyright information.
2. Run the " PatPerl_<version>_<perlVersion>-win32.exe" that came in your distribution and extract the contents into your ActivePerl installation directory (C:\Perl by default). This will install the PatPerl module into the "site" specific portion of the ActivePerl install tree.
3. Copy your "PatPerl.lic.<hostname>" license that came with your distribution into some temporary directory (C:\TEMP, for example).
4. "cd" into that directory and install the license by issuing the command "perl PatPerl.lic.<hostname>" from a MSDOS prompt. This will copy the license file to a good spot under the "site" portion of your Perl installation so the PatPerl module can find it.
5. If you wish, proceed to test that the Module was installed successfully, as outlined in the *Testing the Installation* section that follows.

UNIX

1. Ensure that Active Perl5 or later or Perl compiled in gcc (with multi-threaded option ON) has been installed onto your machine. The path should be properly setup so you can execute "perl -v" from any directory and see the Perl5 version and Copyright information.
2. Copy the distribution tar file into a directory somewhere on the local hard drive and untar it using:

```
tar -xvf PatPerl_<version>5X-<platform>_dist.tar
```

This will extract two files: an installUnixDist.pl script and the distribution body. Install the distribution body using:

```
perl ./installUnixDist.pl
```

You will most likely need to be root to perform the above operation. The PatPerl module files will be installed into the "site_perl" portion of your Perl installation tree.

3. Copy the "PatPerl.lic.<hostname>" license file into some temporary directory (/tmp, for example).
4. "cd" into that directory and install the license by issuing the command "perl PatPerl.lic.<hostname>" from your shell prompt. This will copy the license file to a good spot under the "site" portion of your Perl installation so the PatPerl module can find it.
5. If you wish, proceed to test that the Module was installed successfully, as outlined in the Testing *the Installation* section that follows.

Testing the Installation

First, test that the Module loads and that the license file has been installed correctly. Issue the following from your command-line shell:

```
perl -e "use PatPerl;"
```

This should result in only your prompt being returned to you, meaning the module was installed correctly and the module successfully located the license.

If you get the following error:

```
ERROR: couldn't find license file PatPerl.lic in @INC
path...quitting
```

then your license hasn't been properly installed. Try installing it again, as per the instructions in the installation section above. Contact support@arackal.com if you still have trouble the second time around.

Testing with a PATROL Agent

Several test scripts are included with the Perl Module for PATROL distribution for verifying that the module can interact with your PATROL Agents. This testing is completely optional, but is recommended if you are using the module for the first time. You may also find it useful to examine the test scripts to see how the module functions are used.

To test the functions provided by the PatPerl module, you will need:

1. a PATROL Agent running somewhere in your environment.
2. a username and password for a user allowed to connect to the Agent (use the PATROL Agent's defaultAccount user if possible – see the note for run_test.pl below)
3. At least the PATROLAGENT KM loaded into the Agent (this KM is used for the tests).

Steps for performing the tests:

- On the node where you have installed PatPerl, extract the PatPerl test scripts, also included with your distribution, into a directory of your choice.

The test script bundles are named:

Win2000/WinNT/98/95 – PatPerl_Testers-<version>.exe

UNIX – PatPerl_Testers-<version>.tar.Z

- “cd” into this directory and run the tests by issuing:

```
perl run_all_tests.pl <host> <port> <user> <password>  
OR
```

```
perl getset_test.pl <host> <port> <user> <password>  
perl pconfig_test.pl <host> <port> <user> <password>  
perl event_test.pl <host> <port> <user> <password>  
perl run_test.pl <host> <port> <user> <password> **  
perl UDPRetries_test.pl <host> <port> <user> <password>
```

where:

- <host> and <port> refer to the host and port number that your test PATROL Agents is currently running on.
- <user> and <password> are the username and password for the user that you are connecting to the remote PATROL Agent as.

****NOTE:** the user/passwd combination given here *must* be the user configured in the PATROL Agent's "defaultAccount" configuration variable if you want to run the "run_test.pl" test script. The other scripts will work fine using any userid configured on the machine.

What the test scripts do:

getset_test.pl: performs a variety of object and variable manipulations including setting a namespace variable, getting the value back, getting and setting alarm ranges, and getting application and instance lists.

pconfig_test.pl: performs several tests on PATROL Agent configuration variables to ensure that the pconfig() function works as expected.

event_test.pl: sets event filters, and sends and receives events. This script will return after 60 seconds, even if it hasn't received any events.

run_test.pl: sends the accompanying "test.psl" file over to the remote agent, runs it, and returns the results to the local machine. The “test.psl” file that is run by the PATROL Agent produces a simple history dump for the PAWorkRateExecsMin parameter.

UDPRetries_test.pl: sets the number of retries for UDP connection in PATROL Agent.

run_all_tests.pl: runs all the above tests within this single script.

Each test script (except UDPRetries_test.pl) prints out the number of functions that failed and number that passed.

If you see any failures, it is possible that either the KMs that the scripts assume will be loaded are missing from the PATROL Agent's namespace, or that your PATROL Agent is busy and the functions are timing out. Try checking:

- the customizable variables located near the top of each script to ensure that they are correct for your Agent, or
- try changing the timeout value (set using the `setModuleAttribs()` function)

If these settings appear to be OK, send email to support@arackal.com outlining the issue. It will help support if you can edit the failed script, placing `PatPerl::debug(1)` and `PatPerl::debug(0)` statements around the failed statements, re-run the script, and add the resulting output to the mail message to help speed up the resolution process.

Chapter 3: Using the PatPerl Module

This chapter provides instructions and sample scripts providing an overview as to how to use the PatPerl Module. The samples provided here are also available in electronic format via the web at www.arackal.com/products/.

Loading the module

Perl5 has many built-in functions and operators that make up the language. Extending Perl is relatively painless due to the packaging mechanism that it employs. Because module functions are not a native part of the language, the module must first be imported into the current Perl namespace using Perl's "use" function. The use statement can appear anywhere within the script prior to its first use, but it is recommended that this be one of the first statements in your script.

The following simple example enables PatPerl module debugging.

```
# my first PatPerl script
#
#

# load the module and make its functions available
use PatPerl;

# enable debugging
PatPerl::debug(1);

# done
```

The *debug()* function is prefixed with "PatPerl::" to tell the Perl interpreter that the debug function we want to invoke is in the PatPerl module, not in some other module that may already be loaded. This type of strict function calling must currently be used with the PatPerl module.

Connecting to a PATROL Agent

Interacting with a PATROL Agent involves three simple steps:

- encrypting your password (if it hasn't already been encrypted)
- connecting to the agent and do work
- disconnect from the agent

The encryption offered by the PatPerl module is a special type known as PEM encryption. It is PATROL specific and offers a simple way to keep other users from knowing your clear-text PATROL password. The PatPerl function `encrypt ()` will convert any string into its PEM encrypted equivalent.

The `connect ()` function, used for connecting to a PATROL Agent, returns an integer handle that is used by most of the other PatPerl functions to refer to the given connection. The module can maintain many connections simultaneously, although it is not recommended to exceed several hundred at any given time.

The following example demonstrates how to connect to a remote PATROL Agent.

```
use PatPerl;


# encrypt our password
$host="myPatrolHost";
$port=1987;
$user="patrol";
$pass="myPassword";

$encPass=PatPerl::encrypt($pass);
$handle=PatPerl::connect($host,$port,$user,$encPass);
if($handle==-1)
{
    print("ERROR: couldn't connect to $host:$port as
          $user...quitting.\n");
    exit(-1);
}

# do some work

# disconnect
PatPerl::disconnect($handle);

#done
```

 **Note**
It is good practice to disconnect from the PATROL Agent when you're done to reduce the number of connections maintained by the module

Sample Script #1: Getting the current value of a parameter

To get the value of a parameter, simply connect to the agent and use the `getVar()` function. The following script gets and displays the current value of the `/PATROLAGENT/PATROLAGENT/PAWorkRateExecsMin` variable from the agent running on host "tester".

```
use PatPerl;

# encrypt our password
$host="tester";
$port=1987;
$user="patrol";
$pass="myPassword";

$encPass=PatPerl::encrypt($pass);
$handle=PatPerl::connect($host,$port,$user,$encPass);
if($handle==-1)
{
    print("ERROR: couldn't connect to $host:$port as
          $user...quitting.\n");
    exit(-1);
}
# get the value of the PAWorkRateExecsMin variable
$retval=PatPerl::getVar($handle,
    "/PATROLAGENT/PATROLAGENT/PAWorkRateExecsMin/value");
if($retval eq "" && PatPerl::getLastError() ne "")
{ # got an error - var may not exist
    print("ERROR: couldn't get var value...quitting\n");
    exit(-1);
}
print("Got value of [",$retval,"]\n");

#done
```

Note
Some functions, like `getVar()` return the empty string ("") under normal conditions, so be sure to check that the last error is also "" before assuming that all is well.


```

foreach $appName (@appList)
{ # get version of KM
    $version=PatPerl::getKMVersion($handle,$appName);
    $curAgent=$host.":". $port;
    write; # dump out the report line
}

# done with this agent, so close up
PatPerl::disconnect($handle);
}

#done

```

Running the above script results in a generated report resembling:

```

                    KM Version Report

Agent(port)          KM Name              Version
-----
speed:13333          AGENTWATCH           1.22
speed:13333          BITWATCH             1.2
speed:13333          BITWATCH_CONTAINER   1.3
speed:13333          LOGWATCH             1.2
speed:13333          NT                   1.104
speed:13333          NT_Composites        1.7

```

The KM version information obtained from the PATROL Agent could easily be stored into a file, database, or web page for general consumption.

Sample Script #3: Running a PSL script remotely

There are times when it is useful to execute a chunk of PSL code on a remote agent; to grab parameter histories from the Agent's personal database, for instance. The following sample script uploads a PSL script from the local machine to the remote machine, runs it, and retrieves the results.

Warning!

IMPORTANT NOTE: in order to run a PSL script that you have uploaded to the agent, you must be connected to the agent as its *defaultAccount* user (this is the user that the PATROL Agent normally runs as). If you are connected as any other user, the PSL script will get uploaded to the remote machine, but the agent will not be able to run it due to the permissions that are given to the uploaded file.

Note

It is generally a good idea to use the *defaultAccount* user when dealing with the *getFile()*, *sendFile()*, *runPslFile()*, and *removeFile()* commands.

```
use PatPerl;

$host="tester";
$port=1987;
$user="patrolDefaultAccount";
$pass="patrolPasswd";

# files used for upload/download (local machine)
$uploadfile="getHist.psl";
$dstfile="getHist.psl";

# encrypt our clear-text password (as usual)
$encPass=PatPerl::encrypt($pass);
$handle=PatPerl::connect($host,$port,$user,$encPass);
if($handle==-1)
{ # problem
    print("ERROR: couldn't connect to $host:$port as
$user...quitting.\n");
    exit(-1);
}

# clear the Agent namespace variable that we'll use as a "done" flag
PatPerl::setVar($handle,"/hist_done","0");

# send a file to the remote agent - note the permission bits will be
# set to "000" if we're connected as anything other than the
# defaultAccount user and to "666" otherwise!
print("sending local file: $uploadfile...\n");
$retval=PatPerl::sendFile($handle,$uploadfile,$dstfile,"666");
if($retval==-1)
{ # problem
    print("ERROR: couldn't send file...quitting.\n");
    exit(-1);
}

# run the remote PSL file
print("attempting to run remote psl file...\n");
$retval=PatPerl::runPslFile($handle,$dstfile);
if($retval==-1)
{ # compilation errors
    print("ERROR: compilation errors found in script...quitting\n");
    exit(-1);
}
if($retval==-2)
{ # some general problem - bad user?
```

```

        print("ERROR: couldn't run script...check that user is
defaultAccount.\n");
        exit(-1);
    }

    # wait for our "done" flag to be set to 1 by the getHist.psl script
    while(1)
    {
        $retval=PatPerl::getVar($handle,"/hist_done");
        print("stop variable is: [$retval]...");
        if($retval ne "1")
        { # keep waiting
            print("waiting...\n");
        }
        else
        { # we're done...
            print("done...\n");
            last;
        }
        sleep(5);
    }

    # get the remote file back again
    print("retrieving results file: hist.out...\n");
    $retval=PatPerl::getFile($handle,"hist.out","histout.".time());

    # delete the PSL file we uploaded
    print("deleting the PSL file [".$dstfile."] from the remote
machine...\n");
    $retval=PatPerl::removeFile($handle,$dstfile);
    if($retval== -1)
    { # problem
        print("ERROR: couldn't delete remote file...\n");
    }

    $retval=PatPerl::disconnect($handle);

    #done

```

The above script runs the *getHist.psl* file on the remote agent. *getHist.psl* is a very simple PSL script that simply dumps history, as shown below.

Note
This is PSL code, not Perl code that is run by the remote PATROL Agent.

```

myPort=get("/tcpPort");
myHome=get("/patrolHome");

system(myHome."/bin/dump_hist -p ".myPort." -class PATROLAGENT -inst
PATROLAGENT -param PAWorkRateExecsMin > ".myHome."/remote/hist.out");

set("/hist_done", "1");

```

Getting More Information

For more information on using the Perl Module for PATROL, visit Arackal Digital Solutions' website at <http://www.arackal.com/perl.htm>.

On the site you will find:

- more sample scripts, sample applications, and useful code snippets
- an on-line, HTML based function reference guide
- tips and tricks on using the module and on what things to avoid
- advanced topics
- latest patches and updates
- on-line web support

If you have any questions, comments or suggestions regarding the Perl Module for PATROL, please contact Arackal Digital Solutions at any one of the addresses or phone numbers below. We will be happy to assist you.

Arackal Digital Solutions Inc.
240 Duncan Mill Rd.
Suite 301
Toronto, ON
Canada
M3B 1Z4

T: 416.703.1211
T: 1.877.437.4933
F: 416.703.2544

Support alias: support@arackal.com
Information: info@arackal.com
Web: www.arackal.com