No help available.

Provides quick mouse access to various ActiveState Perl Debugger commands.

Displays short descriptions of menu and tool bar commands, the current line number, and the status of the currently executing Perl program.

Displays Perl expressions that you want to watch while debugging the currently active Perl program.

Displays selected Perl expressions that occur around the instruction pointer.

Displays Perl expressions that you want to watch while debugging any Perl program.

Exits the application. The current debugging session is terminated.

Searches the currently active Perl program for the specified text. You can specify whether or not the search is case sensitive, whether or not you want to match only words, and which direction you want to search.

Searches the currently active Perl program for the next occurrence of the most recently specified text.

Searches the currently active Perl program for the previous occurrence of the most recently specified text.

Jumps to the specified line in the active Perl program.

Adds a non-conditional breakpoint at the currently selected line (or removes any breakpoint already set at that line).

Allows you to add a breakpoint anywhere in the current file with an optional condition.

Displays any breakpoints set in the active Perl program, allowing you to add new breakpoints, edit existing breakpoints, or remove existing breakpoints.

Removes all breakpoints set in the active Perl program.

Sets a bookmark on the current line of the active Perl program; clears the bookmark if one is already set.

Jumps to the nearest bookmarked line below the current line of the active Perl program.

Jumps to the nearest bookmarked line above the current line of the active Perl program.

Removes all bookmarks currently set in the active Perl program.

Shows or hides the tool bar, which contains buttons that allow you to quickly execute various commands.

Shows or hides the status bar at the bottom of the application window.

Shows or hides the watch window, which displays Perl expressions that you want to watch while debugging the currently active Perl program.

Shows or hides the proximity window, which displays selected Perl expressions that occur around the instruction pointer.

Shows or hides the register window, which displays Perl expressions that you want to watch while debugging any Perl program.

Continues execution of the Perl program from the instruction pointer. Execution continues until a breakpoint is hit or the Perl program terminates.

Halts execution of the Perl program, exiting ActiveState Perl Debugger.

Continues execution of the Perl program from the instruction pointer, stopping at the next statement, but stepping into any called subroutines.

Continues execution of the Perl program from the instruction pointer, stopping at the next statement, stepping over any called subroutines.

Continues execution of the Perl program from the instruction pointer, stopping after the current subroutine has returned.

Continues execution of the Perl program from the instruction pointer, stopping when execution reaches the currently selected line.

Ensures that the line pointed to by the instruction pointer is currently visible.

Allows you to quickly evaluate any Perl expression, in the context of the Perl program being debugged.

Displays the contents of the specified variable. Use this command to view the contents of entire arrays or hashes.

Displays the call stack for the instruction pointer. The call stack lists the subroutines that have been called but have not yet returned.

Executes the specified tool. Select "Customize" from the Tools menu to see what application the tool executes.

Customizes the tools listed at the top of the Tools menu. This allows you to run any application right from ActiveState Perl Debugger.

Modifies application settings that control many aspects of its use.

Displays online help for the application.

To obtain help on some portion of ActiveState Perl Debugger, click on the context help button, and then click somewhere in the application window, such as another tool bar button. An appropriate help topic will be displayed.

Displays the version number of this application, as well as copyright and company information.

Adds a watch variable to the window.

Edits the selected watch variable.

Removes the selected watch variable.

The title bar displays the name of the application and/or the name of the active lesson.

Displays a four-headed arrow so that you can size the active window with the arrow keys.

Displays and allows modification of your horizontal position using the mouse.

Displays and allows modification of your vertical position using the mouse.

Displays a four-headed arrow so that you can move the active window with the arrow keys.

Minimizes the active window.

Enlarges the active window to fill all available space.

Switches to the next open lesson window. The order of lesson windows is determined by the order in which they were opened.

Switches to the previous open lesson window. The order of lesson windows is determined by the order in which they were opened.

Exits the application, closes the active lesson window, or cancels the active dialog box, depending on the window.

Restores the size and position of the active window to the state before the most recent Minimize or Maximize command.

Displays a list of all open applications.

Switches to the next pane of the active window.

Switches to the previous pane of the active window.

To close this dialog and save any changes you have made, click OK.

To close this dialog without saving any changes you have made, click Cancel.

To close this dialog, click Close.

Help is available for each item in this group. Click the question mark at the top of the dialog box, and then click the specific item you want information about.

Displays the title and version of your copy of ActiveState Perl Debugger.

Displays the copyright information for ActiveState Perl Debugger.

Displays address information for ActiveState Tool Corp.

Displays the internal build number of ActiveState Perl Debugger executable.

To see the build number for this executable, open the About dialog box with the Control key held down. To then view the credits for ActiveState Perl Debugger, double-click on the icon.

To specify the desired watch expression, type the expression into the Watch box and click OK.

Displays the call stack at the current instruction pointer. The call stack is the list of subroutines that have been called but have not returned yet.

Jumps to the line specified by the currently selected entry in the call stack. This feature is not available if the entry specifies a line in a file other than the currently active Perl program.

Displays the contents of the variable specified in the Variable box after the View button has been clicked. This window actually displays the contents of the specified symbol, so if there is both a scalar and an array of the specified name, both will be displayed.

To view the contents of a variable in the context of the currently active Perl program, type the variable name into the Variable box and click View. The '$', '@', or '%' before the variable name is optional.

Displays the currently active breakpoints and their conditions.

Adds a breakpoint anywhere in the current file with an optional condition.

Edits the line number and condition of the currently selected breakpoint.

Jumps to the line in the active Perl program specified by the currently selected breakpoint.

Removes the currently selected breakpoint.

Removes all currently active breakpoints.

To jump to a specific line number in the active Perl program, type the line number into the Line box and click Go To.

To specify the source line of the breakpoint, type the line number into the Break At Line box.

To specify a condition for the breakpoint, select Condition. Type a Perl expression into the Condition box that returns a true value if and only if you want the breakpoint to stop execution.

To display the colors used by ActiveState Perl Debugger, click on this tab.

To set the background color for the category selected in the Category list, choose a color from the Background list.

To change a color used by ActiveState Perl Debugger, choose a color category from the Category list and then change the Foreground and/or Background lists as desired. The first few color categories are used even when syntax coloring is not enabled.

To set the foreground color for the category selected in the Category list, choose a color from the Foreground list.

To enable syntax coloring of comments, keywords, etc. in the active Perl program, select Enable Syntax Coloring.

To display various options for ActiveState Perl Debugger, click on this tab.

To enable the splash screen at the beginning of ActiveState Perl Debugger session, select Display Splash Screen.

To reset all of the settings used by ActiveState Perl Debugger, click Reset All Settings and restart the debugger.

To display various options relating to the source window of ActiveState Perl Debugger, click on this tab.

To change the font used by the source window, click Set Font.

To specify the number of spaces that equals one tab character, type a number into the Tab Size box.

To display the condition of a breakpoint when the mouse cursor is hovered over the breakpoint mark in the left margin of the source window, select BreakTips.

To display line numbers in the left margin of the source window, select Display Line Numbers.

To display the contents of a variable when the mouse cursor is hovered over the variable in the source window, select WatchTips.

To display various options relating to the watch windows of ActiveState Perl Debugger, click on this tab.

To change the font used by the watch windows (and watch-related dialogs), click Set Font.

To specify the number of lines above the instruction pointer that the proximity window should scan for variables, type a number into the Scan Above IP box.

To specify the number of lines below the instruction pointer that the proximity window should scan for variables, type a number into the Scan Below IP box.

To set the variables in the register window to their default settings, click Reset.

To evaluate a Perl expression in the context of the Perl program being debugged, type the expression into the Evaluate box and click Eval. The result of the Perl expression will appear in the Result box.

To add the Perl expression currently displayed in the Evaluate box to the watch window, click Add Watch.

To send an advanced command directly to ActiveState Perl Debugger, type the command into the box and click OK.

To change the argument string to be passed to the command specified in the Command box, type into the Arguments box. Click the button to the right of the Arguments box to insert special fields into the argument text.

To change the command associated with the selected tool, type the full path of a program executable into the Command box. Click the button to the right of the Command box to browse for the program.

To change the initial directory to be used by the command specified in the Command box, type the full path of a directory into the Initial Directory box. Click the button to the right of the Directory box to browse for the directory.

To change the name of the selected tool, type an appropriate name into the Name box.

Displays the user tools currently available. To modify an existing tool, select the tool name and change the appropriate information. Use the buttons above the list of tools to add a tool, remove a tool, or rearrange the tools.

Adds a new item to the list.

Displays the properties of the selected item.

Moves the selected item down one position in the list.

Moves the selected item up one position in the list.

Removes the selected item from the list.

Credits

Software Development

Edward Ball
Bob Pritchett
Tom Sanocki

Documentation

Edward Ball

Introduction to ActiveState Perl Debugger

Welcome

ActiveState Perl Debugger™ is a compact visual **Perl** debugger designed to replace the default console-based Perl debugger. Experienced programmers will find familiar controls and powerful features, while programmers who are just starting out will find that with its intuitive interface, they will be producing solid code in no time.

We hope you enjoy using ActiveState Perl Debugger.

Key Benefits & Features

ActiveState Perl Debugger™ is a lightweight yet powerful GUI debugger for your perl scripts that lets you focus on the code.

Key Benefits

- **Easy to use**. Just run Perl with the -d option, and you're debugging!
- **Zero Learning Curve**. Whether you are familiar with popular GUI debuggers or not, you will feel right at home with ActiveState Perl Debugger.
- **Saves time**. Remembers your settings, including breakpoints and bookmarks, so that you can focus on your code.

Key Features

- **Proximity Window**. Displays scalar variables near the instruction pointer so you can focus on what is relevant as you step through your code.
- **Syntax Coloring**. Single and double quoted strings, regular expressions, comments, POD, and Perl key words are color coded to help you navigate your script.
- **Detailed Information**. Call stack, list and hash dumps, several watch windows, quick eval and conditional breakpoints let you get the information you need.

About this Documentation

This documentation has been organized into the following sections:

Section 1 – Introduction to ActiveState Perl Debugger™
The **Introduction** section includes information on the following topics:

- the debugger and it's main features
- the documentation and how to navigate it
- contacting ActiveState Tool Corp
- how to purchase the debugger
- additional references in the learning more section

Section 2 – Quick Start
The **Quick Start** section is for users who don't want to read the documentation, but want to get up and running the debugger as quickly as possible

Section 3 – Installation
The **Installation** section is an in depth look at what's happening during the installation process.

Section 4 – Getting Started
The **Getting Started** section includes a brief sample debugging session to illustrate the main features of **ActiveState Perl Debugger™** in action.

Section 5 – Inside ActiveState Perl Debugger™
This section is a more in depth look at all of the features of the debugger**,** including all of its features and how to configure them.

Section 6 – Troubleshooting
The **Troubleshooting** section is there for you should any problems arise while using the debugger. This section currently consists of a list of Frequently Asked Questions (FAQ) and where to go for additional support.

Section 7 – References
The **References** section includes both sub-sections directly related to ActiveState Perl Debugger™ and sections which are there for you should you decide that you want to learn more about **Perl**.

Conventions used in this documentation

- Menu names and options, and important terminology are shown in **bold**.
- Key names, key combinations and key sequences are shown enclosed in brackets and in bold. For example: **<ALT + F1>**
- Source code and commands are shown in `courier`.

Comments on this documentation
If you have any comments or suggestions regarding this documentation, please submit them on the bug reporting page of the ActiveState Web site at: **http://www.ActiveState.com/bugs**.

Contacting ActiveState

To contact **ActiveState Tool Corp.**, please visit the ActiveState Web site at:

- **http://www.ActiveState.com**

Reporting Bugs

To report bugs in **ActiveState Perl Debugger™**, or any other ActiveState product, please visit the bug page on the ActiveState Web site at:

- **http://www.ActiveState.com/bugs**

Purchasing ActiveState Perl Debugger

ActiveState Perl Debugger is available for purchase online at the ActiveState Web site at **http://www.ActiveState.com/pldb/**.

With this release, we are only offering trial and single use licenses.

Trial licenses are freely available and allow full functionality of the debugger. Trials expire after 7 days.

Single use licenses are classified into two separate types: User and Machine.

User Licenses
User licenses are for a specific user. A User license may be used by the same user on any number of machines.

Machine Licenses
Machine Licenses are for a specific machine. A Machine license may only be used on one particular machine, but any number of users may use it.

After this special offered has expired, we will be offering serveral different types of licenses, including Educational, Bulk and Enterprise to suit the needs of the different types of users.

Upgrades & maintenance releases
Your license entitles you to free maintenance releases and revisions. There are several known issues with the debugger, and as we resolve them we will notify you. Upgrading your installation will be a simple matter of installing the new package. Our IDE, Visual Perl is under development, as is a more powerful version of the debugger. Our policy is to give our existing customers a discount on the purchase of our new products when they are released.

For more information on purchasing ActiveState Perl Debugger, and on our licensing system, please see the purchasing and licensing FAQs at: **http://www.ActiveState.com/pldb/faq/**.

Learning More

For more information on **ActiveState Perl Debugger™** and on **Perl** in general, please see to the sections below:

- **Books on Perl**
- **Perl Internet Web sites**
- **Perl related mailing lists**
- **Perl related Usenet newsgroups**

Quick Start

This section is designed to get you up and using **ActiveState Perl Debugger** as quickly as possible. It assumes that you have already gone through the installation process trouble free, and are ready to debug. If you had any problems with installation, please see the **Installation** section.

NOTE: You must have **Perl for Win32™** build 310 or higher installed to use **ActiveState Perl Debugger™**. You can check the version you have installed by typing **`perl –v`** at the command line. The latest version of Perl for Win32 is available from the **ActiveState** Web site at: **http://www.ActiveState.com**.

After successful installation of the debugger, you can start debugging your Perl programs by running Perl with the `–d` option, as follows:

```
Perl –d <myprogram>
```

where **`<myprogram>`** is the Perl program you want to debug.

ActiveState Perl Debugger has all of the basic debugger functionality you expect from a debugger plus a list of additional features designed to make your debugging easier. For a detailed list of the program's features and how to use them, please see the **Inside ActiveState Perl Debugger** section.

Installation

Installation Requirements

Before using **ActiveState Perl Debugger™**, you should be familiar with the basic operation of Windows 95 or Windows NT.

System requirements
ActiveState Perl Debugger was designed to run on a system consisting of:

- Pentium 90 or faster (recommended)
- 16 MB RAM or more
- Windows 95 or Windows NT 4.0
- **Perl for Win32**, Build 310 or later – the latest version is available from the **ActiveState** Web site at **http://www.ActiveState.com** or, Standard Distribution v5.004xx release.
- 2 MB free disk space

Installation Instructions

NOTE: Before you start installing ActiveState Perl Debugger, you should have installed either **Perl for Win32** build 310 or higher – available from the ActiveState Web site at **http://www.ActiveState.com** – or, the Standard Distribution v5.004xx release. To check to see which version of Perl you have, run Perl with the **`–v`** option as follows: **`perl –v`**.

To operate properly, ActiveState Perl Debugger requires an Activation key. To get a key, you will have to go to the ActiveState Perl Debugger Web page at **http://www.ActiveState.com/pldb/default.htm** and select either **Buy-it**, to purchase a key, or **Try-it**, to get a trial key. You can also select **Product licensing** from the **ActiveState Perl Debugger** program group in the **Start** menu, if you have already installed the software. Instructions on how to get your Activation Key will be emailed to you.

To successfully install and activate ActiveState Perl Debugger, you should first run the setup program and then run the Activation Key. The setup program will prompt you (a) to accept the terms of the License Agreement, (b) for the destination directory for ActiveState Perl Debugger, (c) whether you are using the Standard Distribution 5.004xx Release and (d) for the name of a program group to install shortcuts to the help and read me files.

Once ActiveState Perl Debugger has been successfully installed you will have to run the Activation Key to license your application. With the application installed and the Activation Key run, you can start using it immediately by running Perl with the "-d" option, as in:

```
C:\SCRIPTS>perl -d script.pl input.txt output.txt
```

Instead of the console-based debugger that would normally be displayed, a Windows application with the caption "ActiveState Perl Debugger" will appear, displaying the Perl script and pointing to the first line in that Perl script to be evaluated.

Registry entries
During installation, the installer adds the following keys to the registry:

```
[HKEY_CURRENT_USER/SOFTWARE/ActiveState/Perl Debugger/1.0]
[HKEY_CURRENT_USER/SOFTWARE/ActiveWare/Perl5/PERL5DB]
```

The PERL5DB registry key defines which debugger **Perl for Win32** uses. If you want to run both console and visual debuggers, please see the **FAQ** for more information.

If during installation you indicated that you are running the Standard Distribution Release 5.004xx, the following line was added to your `autoexec.bat` file if you are running Windows 95:

```
set PERL5DB = BEGIN { require '<perldbpath>\PerlDB.pl' }
```

where `<perldbpath>` is the path to where you installed the ActiveState Perl Debugger. Users who are running Windows NT will have the Perl5DB variable set in their environment.

Once you have installed the software, you must obtain an Activation Key to allow you to use it. You

NOTE: If for any reason, registration should fail, you can run it again by selecting **Product Licensing** from the **ActiveState Perl Debugger** program group of the **Start** menu.

Getting Started

[Will add tutorial here for final release]

Inside ActiveState Perl Debugger

Can I trap compiler warnings in the debugger? I want to run the debugger with –w –d, but it doesn’t seem to work.

This is a known problem. A useful alternative is to first run the script with –c –w, fix those problems, then run with –d to make sure the script is doing what it should be doing.

The latest Perl Debugger FAQs are always available at: http://www.ActiveState.com/pldb/faq/

I'm trying to run ActiveState Perl Debugger™ using the `-d` option, but I'm getting an error message that says `ole.pm` and `config.pm` cannot be found. What's going on?

It sounds like you don't have Perl properly installed. After you have copied the Perl files onto your machine using the Perl installer, make sure you run the appropriate batch file to complete the install. If the install is successful, the debugger should be able to locate the standard library modules and everything should work fine.

The following registry key contains the complete path to the `/lib` directory:

[HKEY_LOCALMACHINE/Software/ActiveWare/PRIVLIB]

If the directory in this key is not the same as the `/lib` directory's actually location, you should reinstall, as mentioned above, however, changing the registry entry to match your configuration may also fix the problem. (please see your Windows NT documentation for more information on changing registry entries.

The latest Perl Debugger FAQs are always available at: http://www.ActiveState.com/pldb/faq/

I'm using the -d option when I run Perl, but the ActiveState Perl Debugger™ window isn't being displayed, and the console window is displaying a "DB<1>" prompt. What do I do?

If you see the `DB<1>` prompt, you are running the default console-based debugger that comes with Perl. It sounds like your `PERL5DB` environment variable is not being set properly – running the installation program again should fix the problem.

The latest Perl Debugger FAQs are always available at: http://www.ActiveState.com/pldb/faq/

ActiveState Perl Debugger™ **is behaving strangely and I don't know what's wrong. What can I do?**

There are various things that you can try if **ActiveState Perl Debugger™** is not working as you would expect it to.

The first thing you should do is check which version of Perl you are running. If you run Perl with the `-v` option, you can find out which build number you are using. ActiveState Perl Debugger should work with most builds of Perl, but it has only been extensively tested with builds later than 310. Older builds may not work properly.

Another thing you can try is to click **Reset All Settings** in the **Options** dialog of the **Tools** menu. This will cause ActiveState Perl Debugger to forget all of its settings, including window placement, breakpoints, and other settings, but it might solve the problem that you are having.

The latest Perl Debugger FAQs are always available at: http://www.ActiveState.com/pldb/faq/

The splash screen that appears when I start ActiveState Perl Debugger™ **was cool the first dozen times, but how do I get rid of it?**

While ActiveState Perl Debugger is running, select the **Options** command of the **Tools** menu and uncheck **Display Splash Screen** in the **General** tab.

The latest Perl Debugger FAQs are always available at: http://www.ActiveState.com/pldb/faq/

I'm trying to step through my Perl program, but most of the debugging commands are beeping at me. What's going on?

Look at the lower-right hand corner of the **ActiveState Perl Debugger™** window – the status bar should say **Ready** or **Working**. If it says **Working**, the Perl program is currently executing and can't be interrupted by the debugger. You must either wait for your Perl program to stop executing (by hitting a breakpoint or being done with the program) or stop the program manually by hitting **<CTRL + C>** or **<CTRL + BREAK>** in the console window.

The latest Perl Debugger FAQs are always available at: http://www.ActiveState.com/pldb/faq/

I've edited my Perl program and now my breakpoints are all in the wrong places. What do I do?

ActiveState Perl Debugger™ keeps track of breakpoints so that you don't have to enter them every time you run the debugger. However, if you add or remove lines from your program the breakpoints will no longer be in the right places. You can fix them by either selecting **Edit Breakpoints** from the **Edit** menu to change their placement, or remove the breakpoints and add them again in the right places.

The latest Perl Debugger FAQs are always available at: http://www.ActiveState.com/pldb/faq/

Can I use both the console and visual debuggers?

Yes. To the directory where you installed **ActiveState Perl Debugger**, a file called `cmdDB.bat` was copied during installation. Typing `cmdDB` at the prompt in the console window (MSDOS/Command Prompt) will switch to the console debugger for the duration of that console session. If you launch another console session, the default debugger will once again be the ActiveState Perl Debugger.

The latest Perl Debugger FAQs are always available at: http://www.ActiveState.com/pldb/faq/

Does ActiveState Perl Debugger work with the Standard Distribution Release 5.004xx port?

Yes, during installation, specify that you are running the Standard Distribution Release, and your autoexec.bat file will be modified if you are running Windows 95 or, if you are running Windows NT, the PERL5DB variable will be set in your environment.

The latest Perl Debugger FAQs are always available at: http://www.ActiveState.com/pldb/faq/

How can I simulate a CGI environment to debug my scripts in?

ActiveState Perl Debugger™ cannot simulate a CGI environment. However you can use the debugger in your actual CGI environment by using the `-d` option in the CGI call configuration of your Web server. You could also try setting up a new file association specifically for scripts that you want to debug with your Web server, for example, `.pld`.

NOTE: This will only work if you are using `perl.exe`.

The latest Perl Debugger FAQs are always available at: http://www.ActiveState.com/pldb/faq/

Why doesn't syntax coloring work correctly for multi-line strings?

ActiveState Perl Debugger™ does not continue syntax coloring of strings (single and double) or regular expressions across line boundaries. This is a design feature in that there were many circumstances where the syntax coloring can get confused and then the rest of the script would be colored incorrectly. To avoid this, syntax state gets cleared at the end of each line. Visual Perl, an upcoming product, will have more correct syntax coloring.

The latest Perl Debugger FAQs are always available at: http://www.ActiveState.com/pldb/faq/

Why can't I edit scripts directly within the source window?

You cannot edit scripts in **ActiveState Perl Debugger**™ source window. You can configure the debugger to launch your favorite editor by choosing **Customize** from the **Tools** menu and adding your editor. ActiveState is working on an integrated development environment, IDE, for Perl developers called **Visual Perl™** which will include the ability to edit scripts directly within the source window as well as a host of other features that Perl developers have told us they want.

The latest Perl Debugger FAQs are always available at: http://www.ActiveState.com/pldb/faq/

When I try to run my scripts in debug mode, I get the error:

```
Undefined subroutine &Win32::Sleep called at C:\Program Files\ActiveState
Perl Debugger\PerlDB.pl line 153
```

There are two ways to fix this. You can comment out line 153, or upgrade your **Perl for Win32** to the latest build. You can get the latest version of Perl for Win32 from the ActiveState Web site at: **http://www.ActiveState.com**.

The latest Perl Debugger FAQs are always available at: http://www.ActiveState.com/pldb/faq/

FAQ List

My scripts causes a "use of uninitialized value" message when run with perl.exe, but seems to run with no error message in the Debugger.

This is a known issue. The debugger declares variables to put into the proximity window, so the value is “initialized”.

The latest Perl Debugger FAQs are always available at: http://www.ActiveState.com/pldb/faq/

The debugger installed OK, but the licensing part failed, what should I do?

Select **Product Licensing** from the **ActiveState Perl Debugger** program group in the **Start** menu to start the licensing process again. Or, you can select the **Purchase** button in the ActiveState Perl Debugger splash screen.

The latest Perl Debugger FAQs are always available at: http://www.ActiveState.com/pldb/faq/

Why can't I dump a my variable?

My variables cannot be dumped with this version of the debugger. To work around this, assign the variable to a temporary variable using **Quick Eval** and then dump that variable.

The latest Perl Debugger FAQs are always available at: http://www.ActiveState.com/pldb/faq/

How can I buy ActiveState Perl Debugger?

ActiveState Perl Debugger is available online for purchasing at the ActiveState Web site at **www.ActiveState.com**.

The latest Perl Debugger FAQs are always available at: http://www.ActiveState.com/pldb/faq/

API is short for **Application Program Interface**. A good API provide a group of routines, protocols and tools with which programmers can develop a program with greater ease. By doing so, an API provide consistency across applications by providing the same basic tools for all programmers to use. Operating systems such as Windows NT have an API, as do the most popular web servers. (*Also see* ***Win32*****,** ***WSAPI****,* ***ISAPI*** *and* ***NSAPI*** *)*

CGI is short for **Common Gateway Interface**. It was designed to standardize the method in which information is transferred between a web server and an external program. Web servers commonly use CGI programs to get and process information from users. A common example is a form that is first submitted by the user to a web server and then passed to the CGI program via CGI for processing. The main drawback of CGI based programs is the strain they put on server resources, as a new process must be launched every time they are run.

DLL is short for Dynamic Link Library. A DLL is a set of functions that can be executed or data that can be used by a Windows application. Some are specifically for the use of one application, while others, such as many of those that come with Windows 95/NT, can be used by more than one application at the same time.

ISAPI is short for Internet Server API. It is the API for the **Microsoft Internet Information Server** (IIS) Web server. Web applications developed to use ISAPI run considerably faster than those that use CGI, as they are loaded in process with the server.

NSAPI is short for Netscape Server API. It is the API for the **Netscape FastTrack** and **Enterprise** Web servers. Web applications developed to use NSAPI run considerably faster than those that utilize CGI due to a higher level of integration with the Web server.

Perl is short for **Practical Extraction and Report Language**. Designed initially for text processing, Perl has become the most popular language for the writing of CGI programs. This has come about due to the ease with which Perl allows programmers to build and test powerful text processing programs.

PerlEx is a software package from ActiveState™ that utilizes a Web server's API to increase the performance of CGI programs written in Perl.

A **process** is any program that is executing, or “running”.

Win32 is the Windows API on **Microsoft Windows NT** and **Windows 95**.

WSAPI is short for Web Site API. It is the API for the **O'Reilly WebSite Professional** Web server. Web applications developed to use WSAPI run considerably faster than those that use CGI due to a higher level of integration with the Web server.

Perl for Win32™ is ActiveState's port of Perl for Win32 systems. You can download it from the ActiveState Web site at **http://www.ActiveState.com**.

➡ The **Instruction Pointer**, or IP for short, is found in the left-hand margin of the **Source** window. The line that it is pointing to will be executed next. Use the **Step Into**, **Step Over**, **Step Out** or **Run to Cursor** commands to advance the position of the pointer.

▶ The **Current Line Pointer** points to the line which is currently selected. You can change the position of the pointer using the up and down arrows, as well as the **<Home>**, **<End>**, **<Page Up>** and **<Page Down>** keys. If the **<Ctrl>** key is pressed, the window is vertically scrolled without moving the current line pointer.

● The **Breakpoint** icon is shown in the left margin of the **Source** window**.**

🟦 The **Bookmark** icon is shown in the left margin of the **Source** window.

File Menu - Exit

Selecting **Exit** from the **File** menu quits the application. You can also exit by clicking on the close box on the right of the application's title bar, by selecting **Close** from the **System** menu, or by pressing **<Alt + F4>**.

If the debugger is working, the executing Perl program will not automatically terminate, and will probably have to be stopped manually using **<Ctrl+ C>**.

When the application is exited, various application settings are saved to the registry, including:

- All options displayed in the **Options** dialog
- The Size and position of the frame window
- The position of the tool bar
- The size and position of the **Watch** windows
- The position and settings of the **Find** dialog
- The current list of variables in the **Registers** window
- The **User Tools** currently supported

Various settings associated with the current Perl program are also saved, including the list of variables in the **Watch** window, the breakpoints and their conditions and the current bookmarks.

NOTE: If the **Reset All Settings** button of the **Options** dialog has been pressed, instead of saving all of the above settings, the entire Perl Debugger registry key is deleted, forcing all of the debugger settings back to their defaults when the debugger is launched next.

Edit Menu - Find

Selecting **Find** from the **Edit** Menu, or pressing **<Ctrl + F> or <Alt + F3>**, displays the **Find** dialog.

Edit Menu - Go To Line

Selecting **Go To Line** from the **Edit** menu, pressing **<Ctrl +G>,** or double-clicking on the current line indicator in the **status bar**, displays the **Go To Line** dialog.

Edit Menu - Toggle Breakpoint

Selecting **Toggle Breakpoint** from the **Edit** menu or from the **Source** window shortcut menu, or pressing **<F9>** removes any breakpoint set at the current line, or inserts an unconditional breakpoint at the current line if there is no breakpoint currently there.

Edit Menu - Insert Breakpoint

Selecting **Insert Breakpoint** from the **Edit** menu or pressing **<Ctrl + F9>** displays the **Insert Breakpoint** dialog. Clicking the **Condition** check box allows you to enter a condition, which if met, will cause the execution of the program to stop. If the condition is not met, the program will continue to execute until it reaches a valid breakpoint or the program terminates.

Edit Menu - Edit Breakpoints

Selecting **Edit Breakpoints** from the **Edit** menu or pressing **<Alt + F9>** displays the **Edit Breakpoints** dialog.

Edit Menu - Toggle Bookmark

Selecting **Toggle Bookmark** from the **Edit** menu or pressing **<Ctrl + F2>** removes the bookmark at the current line, or sets a bookmark at the current line if there is currently no bookmark there. Double clicking on a line in the margin will also toggle bookmarks.

Edit Menu - Next Bookmark

Selecting **Next Bookmark** from the **Edit** menu or pressing **<F2>**, sets the current line to the next bookmarked line in the script, wrapping to the top of the script if necessary, and beeping if there are no bookmarks in the script.

Edit Menu - Previous Bookmark

Select **Previous Bookmark** from the **Edit** menu or pressing **<Shift + F2>**, sets the current line to the previous bookmarked line in the script, wrapping to the bottom of the script if necessary, and beeping if there are no bookmarks in the script.

Edit Menu - Remove All Bookmarks

Selecting **Remove All Bookmarks** from the **Edit** menu or pressing **<Ctrl + Shift + F2>** removes all bookmarks from the script.

View Menu - Tool Bar

Selecting **Tool Bar** from the **View** menu displays the **Tool Bar** if it is currently hidden, and hides it if it is currently displayed. (The menu item is checked only if the **Tool Bar** is currently shown)

View Menu - Status Bar

Selecting **Status Bar** from the **View** menu displays the **Status Bar** if it is currently hidden, and hides if it is currently displayed (The menu item is checked only if the **Status Bar** is currently shown)

View Menu - Watch

Selecting **Watch** from the **View** menu displays the **Watch** window if it is currently hidden, and hides it if it is currently displayed. (The menu item is checked only if the **Watch** window is currently shown)

View Menu - Proximity

Selecting **Proximity** from the **View** menu displays the **Proximity** window if it is currently hidden, and hides it if it is currently displayed. (The menu item is checked only if the **Proximity** window is currently shown)

View Menu - Registers

Selecting **Registers** from the **View** menu displays the **Registers** window if it is currently hidden, and hides it if it is currently displayed. (The menu item is checked only if the **Registers** window is currently shown)

Debug Menu - Continue

Selecting **Continue** from the **Debug** menu or pressing **<F5>** causes the currently active Perl program to continue executing until it encounters the next breakpoint or until the program is complete. If you do not insert a breakpoint, the program will be run to completion, and the debugger exited.

Debug Menu - Stop Debugging

Selecting **Stop Debugging** from the **Debug** menu, or pressing **<Shift + F5>**, causes the active Perl program to no longer be debugged, exiting the debugger.

Debug Menu - Step Into

Selecting **Step Into** from the **Debug** menu, or pressing **<F11>,** causes the currently active Perl program to execute the next statement. If that statement calls a subroutine, then the current statement advances to the first statement of that subroutine.

Debug Menu - Step Over

Selecting **Step Over** from the **Debug** menu or pressing **<F10>** causes the currently active Perl program to execute the next statement. If that statement calls a subroutine, the execution continues through the subroutine normally (unless a breakpoint is set within that subroutine).

Debug Menu - Step Out

Selecting **Step Out** from the **Debug** menu or pressing **<Shift + F11>** causes the currently active Perl program to continue executing until the **instruction pointer** arrives at the end of the currently executing subroutine. If there is no currently executing subroutine, the command works like Continue.

Debug Menu - Run To Cursor

Selecting **Run to Cursor** from the **Debug** menu or pressing **<Ctrl + F10>** causes the currently active Perl program to continue executing until the **instruction pointer** arrives at the **current line pointer**. The Perl script will run to completion and exit the debugger if execution never reaches the current line pointer.

Debug Menu - Quick Eval

Selecting **Quick Eval** from the **Debug** menu, or pressing **<Ctrl + E>** or **<Shift + F9>** allows you to evaluate a Perl expression in the context of the Perl program currently being debugged. Type the expression into the **Evaluate** box and click the **Eval** button. The result of the Perl expression will appear in the **Result** box. Clicking **Add Watch** will add the expression to the **Watch** window. The result of the expression is always treated as a scalar, even if the expression returns an array or hash. If you need to see the contents of an array or hash, assign the array or hash to a temporary variable, and then use the **Dump Variable** dialog to view its contents.

Any Perl expression can be successfully evaluated with this dialog, including assignments, which can be used to change variables. More specifically, any expression that would be syntactically correct inside parentheses will work with this dialog. As a result, neither blocks nor multiple statements separated by semicolons are supported.

Also, a match or substitution evaluated with this dialog will not affect the global match variables ($&, $1, etc.) if you want to view the results of a match, for example, you'll have to include them in the evaluation in some way, as in "/([a-z]+)/, $1".)

Debug Menu - Dump Variable

Selecting **Dump Variable** from the **Debug** menu or pressing **<Ctrl + D>** displays the **Dump Variable** dialog. This dialog displays the contents of the variable specified in the **Variable** box after the **View** button has been clicked. If there is both a scalar and an array with the same name, both will be displayed.

The Dump Variable dialog displays the contents of a specified variable in the context of the currently executing Perl program. This dialog calls the **dumpvar** subroutine provided by Perl 5.0. The text specified in the **Variable** box can be a scalar, array, hash or even a file handle. The $, @ or % is stripped from the variable before it is passed to **dumpvar**, so if the specified variable name is used by both an array and a hash, for example, the contents of both the array and the hash will be displayed. See the implementation of the **dumpvar** subroutine for a more complete description of what is supported.

Debug Menu - Call Stack

Selecting **Call Stack** from the **Debug** menu or pressing **<Ctrl + T>** displays the **Call Stack** dialog. This dialog displays the call stack, or the currently active subroutines at the **current line pointer**. The currently active subroutines is a list of subroutines that have been called but have not yet returned.

Each entry in the list of subroutine calls includes the name of the subroutine, the arguments that were passed to the subroutine, if any, and the file and line from which the subroutine was called. The **Go To Line** button is enabled when a subroutine call from the currently active Perl program is selected. Pressing the **Go To Line** button, or double-clicking a subroutine call item, will close the **Call Stack** dialog and set the current line to the line number specified by the selected subroutine call in the call stack.

Tools Menu - Edit Source

Selecting **Edit Source** from the **Tools** menu opens the currently active Perl script with **Notepad**. See the **Customize** section for information on how to change the editor and add your own **User Tools**.

Tools Menu - Customize

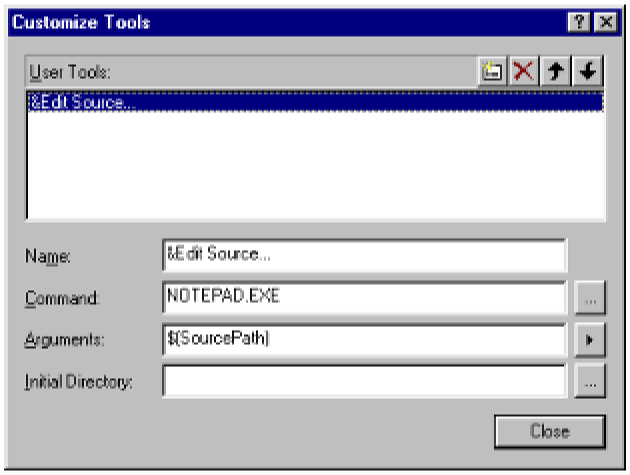
The **Customize Tools** dialog displays the list of user tools currently available, and allows you to add, remove or change the list of **User Tools**. **Selecting Customize** from the **Tools** menu displays the **Customize Tools** dialog. The **Edit Source** option of the **Tools** menu is the default tool and can be configured to launch your favorite editor.

You can add up to 16 tools, and you can run them by either selecting the entry from the Tools menu or pressing **<Alt + 1-9>**, for tools 1 through 9, **<Alt + 0>** for tool 10, and **<Alt + Shift + 1-6>** for tools 11 through 16.

The names of the tools currently available are shown in the User Tools list. Selecting a tool displays the Name, Command, Arguments and Initial Directory for that tool.

The four buttons above the User Tools list allow you to add, remove and change the order of the user tools.

The Add button **<Alt + N>** adds a new tool to the bottom of the list, giving it the name New Tool and no command, arguments or initial directory.
The Delete button **<Delete>** removes the selected item from the list
The Move Up button, **<Alt + Up Arrow>** moves the selected too up one position
The Move Down button, **<Alt + Down Arrow>** moves the selected tool down one position.



Name: &Edit Source...
Command: NOTEPAD.EXE
Arguments: $(SourcePath)
Initial Directory:

To add a new tool:

1. Press **<ALT + N>** or click on the **New** button located in the upper right corner of the dialog.
2. Type the name that you want to appear in the **Tools** menu in the **Name** text box. Use the **&** character to indicate which letter you would like to use for a quick access key
3. Type the name of the application or command that you want run when the menu selection is chosen. You can browse on your computer's disks by pressing the **…** button.
4. Select the arguments you want to append to the command by clicking on the button to the right of the text box, and selecting one of the choices from the shortcut menu. (*see below for an explanation of the arguments you can select*)
5. Choose the initial directory for the tool
6. Press the Close **button** to add the new entry to the **Tools** menu.

The **Arguments** text box displays the arguments that will be passed to the program specified in the **Command** text box when the tool is selected from the **Tool** menu.The following strings with additional information can be passed

Source Path	&(SourcePath)	The full path of the active Perl program, e.g. c:\scripts\buildbook.pl
Source Directory	$(SourceDirectory)	The directory of the active Perl program, e.g. c:\scripts
Source Name	$(SourceName)	The name of the active Perl program, e.g. BuildBook
Source Extension	$(SourceExtension)	The extension of the active Perl program, e.g. .pl
Current Directory	$CurrentDirectory)	The current directory of Windows, e.g. c:\My Documents
Current Line	$(CurrentLine)	The number of the current line in the active Perl program, e.g. 12.

Current Text	$(CurrentText)	The text of the current line of the active Perl program, e.g. chomp;

You can change the order in which the entries appear in the **Tools** menu by selecting the entry in the **Customize Tools** dialog box, and pressing either the up or down arrow located in the upper right corner.

To delete an entry, select the entry and either press the **<Delete>** key, or press the **Delete** button located in the upper right corner of the **Customize Tools** dialog box.

Any changes made take place immediately – there is no way to undo any changes made in this dialog.

Tools Menu - Options

Selecting **Options** from the **Tools** menu displays the **Options** dialog. The Options dialog is composed of four tabs:

- **General**
- **Source**
- **Watch**
- **Colors**

Help Menu - Help Topics

Selecting **Help Topics** from the **Help** menu or pressing **<F1>** displays the ActiveState Perl Debugger online help.

Help Menu - What's This

Selecting **What's This?** from the **Help** menu enables context sensitive help. Subsequently clicking on any part of the main window will display a popup help window briefly describing the clicked item if there is help available for that item.

Help Menu - About Perl Debugger

Selecting **About Perl Debugger** from the **Help** menu displays the **About** dialog.

Tools Menu - Send Command

Pressing **<Ctrl + F7>** allows the user to send a command directly to the PerlDB.pl script used by the debugger. There is no menu item for this dialog, as it is an advanced feature that few, if any people will need to take advantage of.

Menus

- File Menu
- Edit Menu
- View Menu
- Debug Menu
- Tools Menu
- Help Menu

File Menu

- Exit

Edit Menu

- Find
- Go To Line
- Toggle Breakpoint
- Insert Breakpoint
- Edit Breakpoints
- Toggle Bookmark
- Next Bookmark
- Previous Bookmark
- Remove All Bookmarks

View Menu

- Tool Bar
- Status Bar
- Watch
- Proximity
- Registers

Debug Menu

- Continue
- Stop Debugging
- Step Into
- Step Over
- Step Out
- Run to Cursor
- Quick Eval
- Dump Variable
- Call Stack

Tools Menu

- Edit Source
- Customize
- Options

Help Menu

- **Help Topics**
- **What’s This**
- **About Perl Debugger**

Debugger Environment

This section includes information on how to navigate the debugger interface. For more information on specific features of the environment, please see the sections listed below:

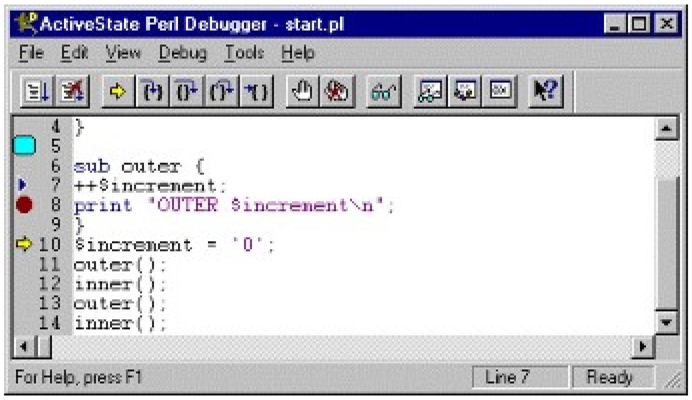
Pointers

Instruction Pointer
The **Instruction Pointer**, or IP for short, is found in the left-hand margin of the **Source** window. The line that it is pointing to will be executed next. Use the **Step Into**, **Step Over**, **Step Out** or **Run to Cursor** commands to advance the position of the pointer.

Current line Pointer
The **Current Line Pointer** points to the line that is currently selected. If you add **Breakpoints** or **Bookmarks**, they will be added to the line this pointer is pointing to. You can change the position of the pointer using the **Up** and **Down Arrows**, as well as the **<Home>**, **<End>**, **<Page Up>** and **<Page Down>** keys. If the **<Ctrl>** key is pressed, the window is vertically scrolled without moving the current line pointer.

Source Window

The source to your Perl program is displayed in the **Source window**. Generally, this is the Perl program specified when you started the Perl debugger, however, it can change if the during the execution of your Perl program, another Perl programs is called. The program is displayed line by line and Perl syntax is emphasized with different text colors if you have enabled syntax coloring in the **Color tab of the Options dialog**.


ActiveState Perl Debugger - start.pl
File Edit View Debug Tools Help
For Help, press F1
Line 7
Ready

```
}

sub outer {
++$increment;
print "OUTER $increment\n";
}
$increment = '0';
outer();
inner();
outer();
inner();
```

To make the source window the active window either click anywhere in the window, or press **<Alt + S>** or **<F6>**.

The left margin displays **breakpoint markers**, **bookmark markers**, the **current line pointer** and the **instruction pointer**. The margin also displays line numbers if you have set this option in the **Source tab of the Options dialog**.

The current line pointer can be moved via the standard keyboard navigation keys: **<Up Arrow>**, **<Down Arrow>**, **<Home>**, **<End>,**   **<Page up>** and **<Page Down>**. If the **<Ctrl>** key is pressed, the window is vertically scrolled without moving the current line pointer.

The current line pointer can also be moved by using the mouse to click on the desired line. If the right mouse button is clicked, or if **<Shift+F10>** is pressed, a shortcut menu is displayed that contains the commands: **Toggle Breakpoint**; **Insert Breakpoint**; **Edit Breakpoints**; **Run to Cursor**; and **Quick Eval**.

Double-clicking on a variable in the **Source window** will select the variable. Right-clicking on the selected variable, produces a shortcut menu from which you can choose **Quick Eval**, **Copy to Watch** or **Copy to Clipboard**.

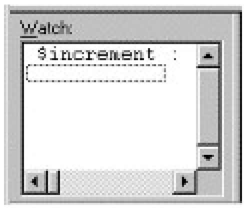
Double-clicking on a line number in the margin will toggle a breakpoint on that line.

Status Bar

On the right side of the **Status Bar** there are two message boxes. **Double-clicking** the message on the left displays the **Go To Line** dialog box. The second message box displays **Ready** when the debugger is idle, and **Working** when a program is running.

Watch Window

The **Watch** window displays one or more Perl expressions in the context of the currently executing Perl program.



The results of each expression in the **Watch** window are updated any time a variable, in the context of the Perl program, could change – after execution is continued, a statement is stepped over, or even after the **Quick Eval** dialog is used. The expressions supported are the same types of expression supported by the **Quick Eval** dialog, and since they can actually change the contents of the variables, it is important to note that the expressions are evaluated from top to bottom. Also note that expressions in the **Watch** window are evaluated before expressions in the **Registers** window.

To make the **Watch** window the active window, either click anywhere in the window or press **<ALT + W>**.

To add variables to the **Watch**, use any of the following methods:

- Right-click in the **Watch** window and select **Add Watch** from the shortcut menu
- Double click on or below the entry at the bottom of the list in the **Watch** window
- Press **<Enter>** when the **Watch** window is the active window, and the entry at the bottom of the list is highlighted (pressing **<Enter>** with a different expression highlighted allows you to edit that particular expression)
- Press the **<Insert>** key while the **Watch** window is active
- Type any character when the **Watch** window is the active window. Any typed characters are automatically added to the **Watch** box when it is displayed.

You can also add variables to the **Watch** window by selecting the variable you want to add in the **Source** window, right-clicking on it, and then selecting **Copy to Watch** from the shortcut menu.

You can **Edit** a watch by doing any of the following:

- Select **Edit Watch** from the shortcut menu of the **Watch** window
- Double-click on an existing watch expression in the **Watch** window
- Press the **<Enter>** key when an existing watch expression in the **Watch** window is highlighted

You can **Remove** a watch by any of the following ways:

- Right-click on the watch expression you want to remove, and select **Remove Watch** from the shortcut menu of the **Watch** window.
- Select the watch expression you want to remove, and press the **<Delete>** key.

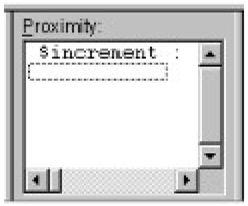
Right-clicking in the border of the **Watch** window displays a menu that allows you to:

- **Allow docking** of the window
- **Hide** the window

You can move the **Watch** window to any position on your screen by clicking on the border of the window and dragging it to the desired location. To prevent the window from docking, keep the **<Ctrl>** key pressed while dragging the window.

Proximity Window

The **Proximity** window looks and behaves much like the **Watch** and **Registers** windows, except that the user cannot change the list of expressions. Instead, the list is automatically generated by the debugger, which looks at the lines of the Perl program surrounding the current statement. You can specify the number of lines above and below the **instruction pointer** to scan for variables by changing the **lines above** and **lines below** options in the **Watch** tab of the **Options** dialog. (**Tools** menu)



To make the **Proximity** window the active window, either click anywhere in the window or press **<ALT + P>**.

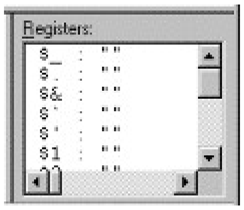
Right-clicking in the **Proximity** window displays a shortcut menu that allows you to:
- **Hide** the **Proximity** window
- **Copy to watch** – copy the currently selected variable to the **Watch** window

Right-clicking in the border of the **Proximity** window displays a shortcut menu that allows you to:
- **Allow docking** of the window
- **Hide** the window

You can move the **Proximity** window to any position on your screen by clicking on the border of the window and dragging it to the desired location. To prevent the window from docking, keep the **<Ctrl>** key pressed while dragging the window.

Registers Window

The **Registers** window looks and behaves exactly like the **Watch** window. The only significant difference between the two is that the expressions listed in the **Registers** window are tied to the application instead of the currently active Perl program, making them the same for every Perl program you debug. Any changes made to the list of expressions in this window will be reflected the next time that the debugger is run, regardless of the script.



```
$_ : ""
$. : ""
$& : ""
$` : ""
$' : ""
$1 : ""
```

The results of each expression in the **Registers** window are updated any time a variable, in the context of the Perl program, could change – after execution is continued, a statement is stepped over, or even after the **Quick Eval** dialog is used. The expressions supported are the same types of expression supported by the **Quick Eval** dialog, and since they can actually change the contents of the variables, it is important to note that the expressions are evaluated from top to bottom. Also note that expressions in the **Watch** window are evaluated before expressions in the **Registers** window.

To make the **Registers** window the active window, either click anywhere in the window or press **<ALT + W>**.

To add variables to the **Registers**, use any of the following methods:

- Right-click in the **Registers** window and select **Add Watch** from the shortcut menu
- Double click on or below the entry at the bottom of the list in the **Registers** window
- Press **<Enter>** when the **Registers** window is the active window, and the entry at the bottom of the list is highlighted (pressing **<Enter>** with a different expression highlighted allows you to edit that particular expression)
- Press the **<Insert>** key while the **Registers** window is active
- Type any character when the **Registers** window is the active window. Any typed characters are automatically added to the **Watch** box when it is displayed.

You can **Edit** a watch by doing any of the following:

- Select **Edit Watch** from the shortcut menu of the **Registers** window
- Double-click on an existing watch expression in the **Registers** window
- Press the **<Enter>** key when an existing watch expression in the **Registers** window is highlighted

You can **Remove** a watch by any of the following ways:

- Right-click on the watch expression you want to remove, and select **Remove Watch** from the shortcut menu of the **Registers** window.
- Select the watch expression you want to remove, and press the **<Delete>** key.

Right-clicking in the border of the **Registers** window displays a menu that allows you to:

- **Allow docking** of the window
- **Hide** the window

You can move the **Registers** window to any position on your screen by clicking on the border of the window and dragging it to the desired location. To prevent the window from docking, keep the **<Ctrl>** key pressed while dragging the window.

Toolbar

The **ActiveState Perl Debugger™** Toolbar conveniently groups in one place the most commonly used functions of the debugger. The function of each of the icons and their corresponding shortcut keys are listed below.

Continue <F5> Continues execution of the Perl program from the Instruction Pointer Execution continues until a break point is hit, or the program terminates.

Stop Debugging <SHIFT+F5> Halts execution of the Perl program, exiting ActiveState Perl Debugger.

Show Next Statement <ALT+NUM *> Ensures that the line pointed to by the Instruction Pointer is currently visible.

Step Into <F11> Continues execution of the Perl program from the Instruction Pointer, stopping at the next statement, but stepping into any subroutines called.

Step Over <F10> Continues execution of the Perl program from the Instruction Pointer, stopping at the next statement, stepping over any called subroutines.

Step Out <SHIFT + F11> Continues execution of the Perl program from the Instruction Pointer, stopping after the current subroutine has returned.

Run to Cursor <CTRL + F10> Continues execution of the Perl program from the Instruction Pointer, stopping when execution reaches the **Current Line Pointer**.

Insert Breakpoint <CTRL + F9> Allows you to enter a breakpoint at an optional condition anywhere in the current file.

Remove All Breakpoints <CTRL + SHIFT + F9> Removes all breakpoints set in the active Perl program.

Quick Eval <SHIFT + F9> Allows you to quickly evaluate any Perl expression, in the context of the Perl program you are debugging.

Toggle Watch Shows or hides the Watch window, which displays Perl expressions that you want to watch while debugging the currently active program.

Toggle Proximity Shows or hides the Proximity window, which displays Perl expressions that occur around the **Current Line Pointer**.

Toggle Registers Shows or hides the Registers window, which displays Perl expressions you want to watch while debugging any Perl program.

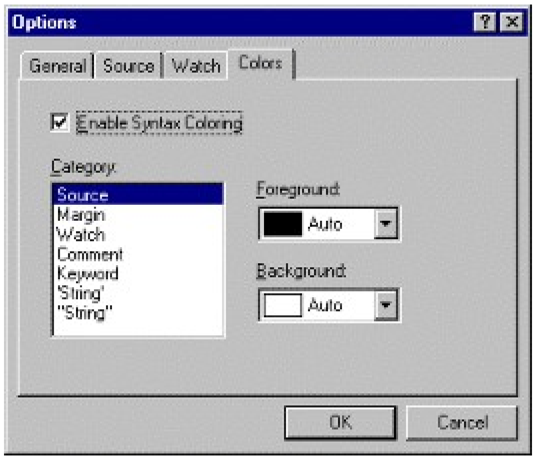
Help To obtain context sensitive help for ActiveState Perl Debugger, first click on the help button, and then click on the item for which you want help and an appropriate help topic will be displayed.

Configuring

This section includes information on customizing how the debugger looks and works.

- **Options Dialog – General Tab**
- **Options Dialog – Source Tab**
- **Options Dialog – Watch Tab**
- **Options Dialog - Colors Tab**
- **Customize Tools**

Options Dialog – Colors Tab



Select **Options** from the **Tools** menu and then click on the **Colors** tab to view the configurable color properties.

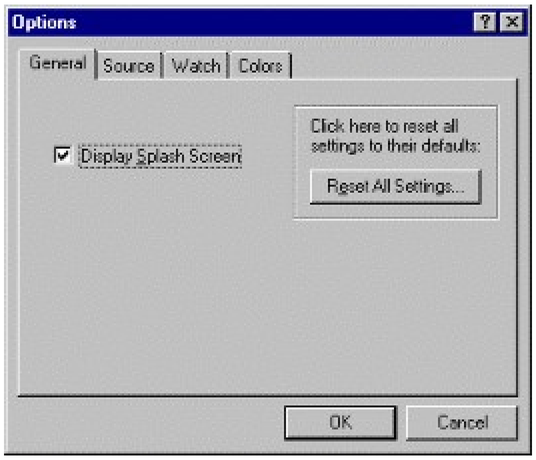
Enable Syntax Coloring

Syntax Coloring allows you to set colors for which various categories of syntax in the Perl program are colored. The components are divided into the following categories:

- **Source** for any Perl source code not modified by any of the categories below
- **Margin** for the margin along the left side of the source window. The foreground color specifies the color of the line numbers, while the background color specifies the color of the margin itself.
- **Watch** sets the foreground and background colors for the **Watch**, **Proximity** and **Registers** windows.
- **Comment** sets the color for any comments that are contained in your source code. One-line comments start with a **#** character not immediately following a **$** character and end at the end of the line. Multi-line comments start with an equal sign followed by an alphabetic character and end with a line that begins with **=cut**.
- **Keyword** sets the color for any keywords in your source code. A complete list of keywords is contained in the file `<perldbpath>/keywords.txt`, where `<perldbpath>` is the path to where you have installed ActiveState Perl Debugger. You can edit the `keywords.txt` file to add or remove any keywords you want. Lines beginning with a semicolon in the file are ignored.
- **'String'** determines the color that **single-quoted strings** are displayed in. Single quoted strings start with a non-backslashed single-quote and end with a non-backslashed single-quote. Single-quoted strings can also start with a q not immediately preceded by a letter, number, or variable prefix. The character immediately following the q is used to determine where the string ends – the next non-backslashed instance of that character terminates the string (unless the character is a bracket, in which case the next non-backslashed closing version of the bracket terminates the string.
- **"String"** determines the color that **double-quoted strings** are displayed in. Double-quoted strings can start with a non-backslashed double-quote and end with a non-backslashed double-quote. Double-quoted strings can also start with qq – the syntax is the same as q for single-quoted strings.

While you are editing any of the options in the **Options** dialog, clicking **OK** at any time accepts and applies the changes you've made. Clicking **Cancel** at any time discards any changes you've made. Clicking the **Reset All settings** button in the General tab will return all of the settings to their defaults.

Options Dialog – General Tab



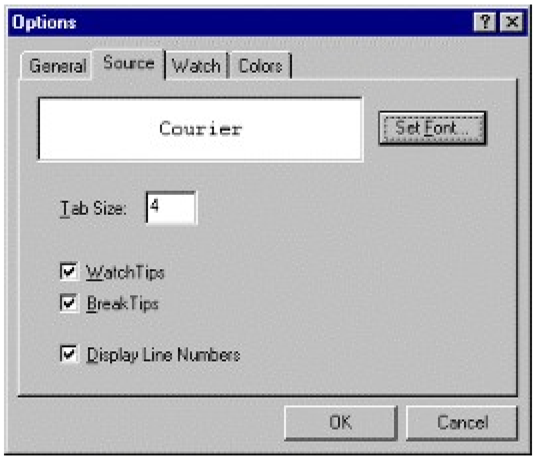
Select **Options** from the **Tools** menu and then click on the **General** tab to view general configurable properties.

Display Splash Screen
The **Display Splash Screen** check box controls whether or not to display the splash screen when you start ActiveState Perl Debugger.

Reset All Settings
Reset All Settings button to discard any changes you have made to the settings, including window placement and breakpoints. This will delete settings not only for the current script, but any other scripts you have run. (Settings are remembered for up to 100 scripts)

While you are editing any of the options in the **Options** dialog, clicking **OK** at any time accepts and applies the changes you've made. Clicking **Cancel** at any time discards any changes you've made.

Options Dialog – Source Tab



Select **Options** from the **Tools** menu and then click on the **Source** tab to configure how you view your source.

Set Font
Clicking the **Set Font** button will bring up a second dialog box from which you can select the font you would like to use to display Perl source code and the line numbers in the left margin. The default is 10 point Courier.

Tab Size
Tab size allows you to specify the number of spaces that represent one tab character in the source window. The default is 4.

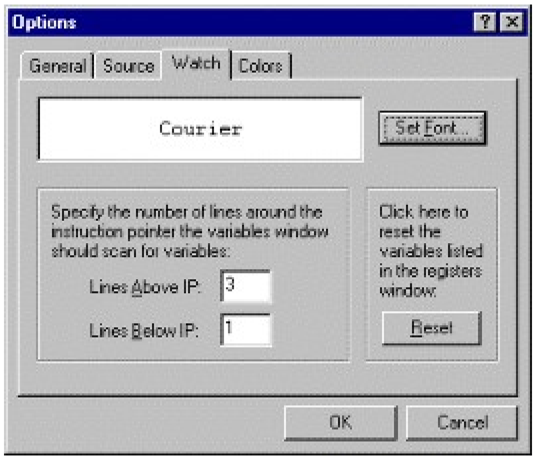
WatchTips
The **WatchTips** check box determines weather or not WatchTips are displayed. With WatchTips enabled, pointing with the mouse to a variable in the source code displays the contents of that variable in a ToolTip.

BreakTips
The **BreakTips** check box determines whether or not BreakTips are displayed. With BreakTips enabled, pointing with the mouse to a breakpoint in the left margin of the source code displays the condition of the breakpoint in a ToolTip. If there is no condition, the ToolTip displays **Break always**, otherwise it displays **Break when <condition>**, where **<condition>** is the condition you have set for that breakpoint.

Display Line Numbers
The **Display Line Numbers** check box controls whether or not to display line numbers in the left side margin.

While you are editing any of the options in the **Options** dialog, clicking **OK** at any time accepts and applies the changes you’ve made. Clicking **Cancel** at any time discards any changes you’ve made. Clicking the **Reset All settings** button in the **General tab** will return all of the settings to their defaults.

Options Dialog – Watch Tab



Specify the number of lines around the instruction pointer the variables window should scan for variables:

Lines Above IP: 3

Lines Below IP: 1

Click here to reset the variables listed in the registers window.

Select **Options** from the **Tools** menu and then click on the **Watch** tab to view the configurable watch properties.

Set Font

Clicking the **Set Font** button will bring up a second dialog box from which you can select the font you would like to use to display text in the **Watch**, **Proximity** and **Registers** windows, as well as the **Quick Eval** and **Dump Variable** dialogs. The default font is 10 point Courier.

Reset

Clicking on **Reset** resets the variables listed in the **Registers** window, returning them to their default values. Default values for the Registers window are: $_, $., $&, $`, $R', and $1 through $9. The values will be reset once you have clicked the **Reset** button.

Lines Above IP

Lines Above IP specifies how many lines above the **instruction pointer** will be scanned for variables to display in the **Proximity** window.

Lines Below IP

Lines Below IP specifies how many lines below the **instruction pointer** will be scanned for variables to display in the **Proximity** window.

While you are editing any of the options in the **Options** dialog, clicking **OK** at any time accepts and applies the changes you've made. Clicking on **Cancel** at any time discards any changes you've made. Clicking the **Reset All settings** button in the **General tab** will return all of the settings to their defaults.

Troubleshooting

- Frequently Asked Questions
- Support

Support

If you’re having a problem with the ActiveState Perl Debugger, your first line of defense is the Frequently Asked Question (FAQ) section. In this section, we have solutions to a list of common problems from users like you. You can find the most up to date list of FAQs for ActiveState Perl Debugger on our Web site at: **http://www.activestate.com/pldb/faq/**.

Peer level support can be obtained from the ActiveState Perl Debugger mailing list, please see the mailing list section for more information on how to join this list. Also, support for installation related problems is available by emailing **Support@ActiveState.com**.

Commercial level support for any of your Perl problems is available from The Perl Clinic. For more information regarding the Perl Clinic and its services, please visit their Web site at **http://www.PerlClinic.com**.

FAQ

Listed below are questions frequently asked about **ActiveState Perl Debugger™.** Click on any one of them for the answer to that particular question.

Reference

Glossary

- API
- Bookmark icon
- Breakpoint icon
- CGI
- Current Line Pointer
- DLL
- Instruction Pointer
- ISAPI
- NSAPI
- Perl
- PerlEx
- Perl for Win32
- Process
- Win32
- WSAPI

Other Perl Tools

ActiveState Tool Corp.™ makes a wide range of **Perl** related software for **Win32** systems. All of the software listed below is available from the **ActiveState** Web site at **http://www.ActiveState.com**.

- **Perl for Win32™**. The de facto standard Win32 port of Perl.
- **PerlEx™**. PerlEx is ActiveState's answer for slow running Perl CGI programs. Compatible with most popular Windows NT Web servers, including **O'Reilly WebSite Pro**, **Microsoft Internet Information Server** and **Netscape Fastrack** & **Enterprise** servers, PerlEx functions similar to **mod_perl** by precompiling your Perl CGI programs for lighting fast execution. PerlEx also adds increased functionality through the introduction of a new powerful type of interpreter called Solar.
- **PerlScript™**. An **ActiveX** scripting engine compatible with **Microsoft Internet Information Server** versions 3.0 & 4.0, **Microsoft Internet Explorer** versions 3.0 & 4.0, **Microsoft Exchange** 5.5 and other ActiveX scripting hosts.
- **Perl for ISAPI™**. A DLL that offers increased performance for Perl programs running on **Microsoft Internet Information Server** and other Web servers that support ISAPI.

Mailing Lists

ActiveState maintains the following mailing lists for it's commercial Perl products

- **Perl-Debugger**. This list is for any questions regarding ActiveState Perl Debugger™, ActiveState's commercial visual debugger for Perl.
- **PerlEx**. This list is for any questions regarding PerlEx, ActiveState's commercial software package for running Perl CGI programs on Windows NT Web servers.

There are also seven mailing lists maintained by ActiveState for the Perl for Win32 community:

- **Perl-Win32-Users**. This list is for general questions on Perl installation on Win32 platforms (except HTTP servers), Cross-platform issues related to Win32, Socket programming (non-Internet), Object Linking and Embedding, COM or Automation (except database related). As well as, Any Perl for Win32 topic not mentioned in the topic lists of the other mailing lists. Please check that FAQs on the ActiveState Web site and related sites before asking a question. This list gets to have some moderate traffic at times.

- **Perl-Win32-Porters**. This list is **NOT** for usage or install questions. If you post a question like that on this list, people may not be nice to you. It is used to discuss porting and development issues. Relatively low traffic that gets little bursts on occasion. Perl5-porters is on this list so that the main porters know what the Win32 issues are.

- **Perl-Win32-Announce**. This list is a receive-only list used to announce new builds, security issues and bugs found and other occasional information. This list can only be posted to by the owner and is very light traffic wise. Any serious **Perl for Win32** developer should be on this list. :-)

- **Perl-Win32-Web**. This list is for the discussion of CGI programming, **PerlIS**, Installation/Configuration of Perl on Win32 HTTP servers, Internet system security related to Perl, **PerlScript** on Web servers, All other Perl/Web related topics, e.g., Web search scripts, HTML munging.

- **Perl-Win32-Admin**. This list is for the discussion of the use of Perl in user account creation, modification, etc. registry work, event logging, Access Control Lists (permissions), and all other systems administration topics.

- **Perl-Win32-Database**. This list is for the discussion of Open Database Connectivity issues, Module installation, Database access via OLE, and all other database related topics.

- **Perl-Win32-J**. This list is for users in Japan who are using Perl for Win32. Discussions are all in Japanese.

To subscribe to any of the above mailing lists, send an email request to **ListManager@ActiveState.com**, with the following in the message:

```
subscribe <listname>
```

Where **`<listname>`** is the name of the list you want to subscribe to.

Copyright Info

Internet Web sites

- **ActiveState Tool Corp** – http://www.ActiveState.com
- **The Perl Clinic** – http://www.PerlClinic.com
- **The O'Reilly Perl Resource Center** – http://perl.oreilly.com/
- **Books on Perl published by O'Reilly** – http://www.oreilly.com/publishing/perl/
- **The Perl Journal** – http://www.tpj.com/
- **The Perl Language Home Page** – http://www.perl.com/perl/

Usenet Newsgroups

Usenet newsgroups are another good source of information and advice regarding **Perl**. Please consult your system administrator or Internet provider for availability of the following groups:

- **comp.lang.perl.announce**
- **comp.lang.perl.misc**
- **comp.lang.perl.modules**
- **comp.lang.perl.tk**

There are also some non-English Perl related newsgroups:

- **de.comp.lang.perl**
- **fj.lang.perl**
- **fr.comp.lang.perl**
- **it.comp.lang.perl**
- **pl.comp.lang.perl**
- **japan.comp.lang.perl**

The Perl Clinic™

The Perl Clinic offers commercial level support for all of your Perl problems. For more information regarding the Perl Clinic and its services, please visit their Web site at **http://www.PerlClinic.com**.

Registry entries

During installation, the setup application made **ActiveState Perl Debugger™** your default debugger by adding the following registry key:

```
[HKEY_CURRENT_USER/Software/ActiveWare/Perl5/Perl5DB]
```

The value was set to:

```
BEGIN { require ‘<perldbpath>\PerlDB.pl’ }
```

where `<perldbpath>` is the path to where **ActiveState Perl Debugger**™ is installed on your system.

To switch back to the Console debugger, change the value to:

```
SET PERL5DB=BEGIN { require '<perlpath>\lib\Perl5DB.pl' }
```

where `<perlpath>` is the path to where **Perl for Win32** installed.

You can temporarily use the console debugger by running the `cmdDB.bat` file located in the directory that you installed ActiveState Perl Debugger in. If you quit that particular console session, the default bugger will revert to the visual debugger.

The **Standard Release 5.004xx** port does not currently use the Perl5DB registry key, requiring you to set the Perl5DB variable as follows:

```
SET PERL5DB=BEGIN { require ‘<perldbpath>\perlDB.pl’ }
```

For more information regarding how to use both the console and visual debuggers and on getting ActiveState Perl Debugger working with the Standard Distribution 5.004xx release, please see the FAQ section.