

User Guide:

Zend Guard 5.0



By Zend Technologies, Inc.

Disclaimer

The information in this document is subject to change without notice and does not represent a commitment on the part of Zend Technologies Ltd. No part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or information storage and retrieval systems, for any purpose other than the purchaser's personal use, without the written permission of Zend Technologies Ltd.

All trademarks mentioned in this document, belong to their respective owners.

© 1999-2007 Zend Technologies Ltd. All rights reserved.

Zend Guard™ User Guide issued December 2007

Product Version: Zend Guard™ 5.0.1

DN: ZG-UG-120407-5.1-004

Contribute to the Documentation

Your feedback is important to us. Therefore, at the bottom of each page is a link for sending e-mails directly to the Zend documentation team.

You can also let us know your views on Zend documentation by participating in our [Zend Documentation Satisfaction survey](#).

Table of Contents

Zend Guard	6
Contribute to the Documentation	7
Introduction	8
Installing Zend Guard.....	9
Launching Zend Guard	11
Registering Zend Guard.....	12
Zend Guard Features	14
Workspace Overview.....	16
Menu Options.....	21
Working with Zend Guard.....	25
Getting Started	26
Modifying Project Contents.....	30
Basic Tutorial	32
Preferences	35
Overriding Project Preferences.....	37
Configuring the Integration with Zend IDE	38
Importing an Existing Project	39
Path Variable Preferences	41
Editing Path Variables	42
Tabs	44
Overview Tab.....	45
Security Tab	48
Exclude Tab	51
Excluding PHP Entities	55
Header Tab.....	58
XML Editing Tab	60
Encoding	62
Encoding	62
How to Encode	64
Distributing Encoded Files.....	65
Manual Encoding	65
Obfuscation	66
Encoding Only	68
Variable Obfuscation	68
Function Obfuscation.....	68

Testing and Debugging Applications	69
Licenses	69
Creating a License	69
License Files	78
Installing a License File	79
License Enforcement	80
License File Location	81
Zend Host ID	82
License Architecture and Behavior	84
Command Line	87
zendenc and zendenc5 – Command Line	87
zendenc - Command Line Examples	92
Creating a Signature License (Command Line)	93
zendenc_sign – Command	96
Zend Guard API	98
FAQ	102
Index	107

Zend Guard

Disclaimer

The information in this document is subject to change without notice and does not represent a commitment on the part of Zend Technologies Ltd. No part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or information storage and retrieval systems, for any purpose other than the purchaser's personal use, without the written permission of Zend Technologies Ltd.

All trademarks mentioned in this document, belong to their respective owners.

© 1999-2007 Zend Technologies Ltd. All rights reserved.

Zend Guard™ User Guide issued December 2007

Product Version: Zend Guard™ 5.0.1

DN: ZG-UG-120407-5.1-004

Contribute to the Documentation

Your feedback is important to us. Therefore, at the bottom of each page is a link for sending e-mails directly to the Zend documentation team.

You can also let us know your views on Zend documentation by participating in our [Zend Documentation Satisfaction survey](#).

Introduction

Zend Guard™ is the first Electronic Licensing solution for the PHP marketplace. It includes the Encoding solution that pioneered PHP intellectual property protection. Unprotected intellectual property, in the form of plain text PHP scripts and software without license restrictions, can be copied, modified, and retained by someone else. It is available to your competitor, to hackers and even to developers at customer sites.

Zend Guard™ provides tools that significantly lessen risk to your intellectual property. It is designed to prevent your property from being viewed or modified. Software piracy losses run in the billions. Estimates place it at \$59 billion in the last five years. This translates into lost opportunities and diminished sales.

Zend Guard maximizes software profitability by:

- Limiting unauthorized duplication or use of your applications.
- Ensuring that only licensed customers use your products and that they remain within the terms of your license.
- Offering flexible licensing terms that make your software products more attractive, increase sales, and improve customer satisfaction.
- Increasing conversion rate from evaluation to licensed product.
- Preventing other people from changing your code (all files are rendered as un-editable and external changes will corrupt the code) protecting the files against external tampering.

Installing Zend Guard

Installation is initiated either by downloading the product from zend.com or by the Zend IDE during its installation (it will install from the web).

Regardless of installation method, Zend Guard provides a standalone solution for encoding files as well as electronic licensing for protecting intellectual property.

System Requirements

To view the Zend Guard system requirements, go to:

http://www.zend.com/products/zend_guard/system_requirements

Please note that some requirements may change from time to time.

Version Compatibility

- Encoded files which have obfuscated **local** variables are only compatible with Zend Optimizer version **2.6** and above.
- Encoded files which have obfuscated variables and also have obfuscated **functions, classes** and/or **PHP Internal symbols** require the **latest version** of Zend Optimizer (PHP4 Optimizer 3.0 and above; PHP5 Optimizer 3.3 and above).
- Encoded files can be used with PHP 4.3 or greater.
- Obfuscation of entities other than local, requires that the latest version of Zend Optimizer be installed. The Zend Optimizer is available for download *free of charge*, from the Zend Store, at: <http://www.zend.com/store>.
- PHP files encoded with Zend Guard 4.0 require Zend Optimizer 3.0 (or greater) in order to run. Encoded files can be used with PHP 4.3 (or greater).

Note:

Newer PHP language features and constructs should be run on PHP versions that support them. This is true for both encoded and non-encoded PHP files.

Extracting the Standalone Zend Guard

- **Windows** - Double-click on self-extracting archive, and follow the instructions in the installation wizard.
- **Linux** - Extract the downloaded tar.gz file: **ZendGuard-5_0_x.tar.gz**. As soon as the extraction is done, run the following extracted binary: **./ZendGuard-5_0_x.bin** and follow the instructions in the installation wizard.
- **Mac** - Extract the zip file double clicking on it. Double click on the archive directory in order to start the Zend Guard installation executable and follow the instructions in the installation wizard.

Launching Zend Guard

Depending on the platform you are using, users can launch the Zend Guard:

- In a new Program Group
- In an existing Program Group
- In the Start Menu
- On the Desktop
- In the Quick Launch Bar
- Home directory

The choices determine how and from where you will be able to activate the Zend Guard.

Zend IDE users can activate Zend Guard from the Zend IDE User interface (**Tools | Encode Project**) or from the Shortcut Menu.

Installing Zend Guard via the Zend IDE installation adds additional integration options. The integration enables users to open and edit encoded Files with Zend IDE, directly from the Zend Guard.

Note:

These options are available even if the installation is not done the same time. The only difference is that the integration settings are automatically configured whereas; in separate installations these options have to be manually configured.

Installing the Zend Guard via the Zend IDE actually starts a separate download and installation process.

- The Zend IDE Installer prompts the user to select to download the Zend Guard or not.
- Selecting to install Zend Guard, adds a download request to the Zend IDE Installer.
- Confirming the download request (clicking the download button) runs two actions:
 1. Starts to download the Zend Guard package from zend.com
 2. When the download is completed, automatically runs the Zend Guard installation process.

Registering Zend Guard

This procedure describes how to register your Zend Guard product. PHP files can be encoded with an unregistered product however, encoded files will **not be optimized** and will be valid for 14 days, only.

License files that are generated with an evaluation version will be valid for 3 days, only.

These limits provide sufficient time to fully evaluate all the Zend Guard features in a fully functioning environment and test the encoding capabilities.

In order to benefit from full Zend Guard capabilities the product should be registered.

Before registering your product make sure you have your registration information or license file available. A license file is a file that contains product activation information. Without this file the registration process will not succeed.



To Register Zend Guard:

1. Go to Help | Register.

The registration Dialog will open.



Figure 1 - Registration Dialog

2. Choose one of the registration options:

- **Download a license file.** Obtain a license file from **www.zend.com**.

Select this option:

- If you want to download a license file from www.zend.com

- If you do not have a local copy of a license file on your computer.
- Registration Dialog Options
 - **Username** - Your zend.com user name (not your e-mail).
If you do not remember your Zend user name please contact sales@zend.com". You can also create a license file from your Pickup Depot,
<https://www.zend.com/store/pickup.php>.
 - **Password** Your zend.com password
 - **Serial Number** the product purchase serial number as it appears in the pickup depot,
<http://www.zend.com/store/pickup.php>
- **Search for License file.** Select this option, if you have a local copy of a license file on your computer. You should register the product by using a *local copy* of a license file when reinstalling the product (or if a license file was sent directly to you by a Zend representative).

Your product will now be registered and all validation restrictions will be removed. If the restrictions persist or if the license is not accepted, please refer to our [Support Center](#) for further information and assistance.

Zend Guard Features

Zend Guard is the industry standard in PHP intellectual property protection. It protects exclusive and commercial PHP applications by obfuscating PHP scripts. Developers enjoy the benefits of a leading open-source language while protecting PHP code when ready to distribute applications. By protecting applications, Independent Software Vendors (ISVs) expand distribution and revenue on maintenance and support.

Zend Guard includes three features to provide a double layer of protection for PHP applications:

- **Encoding**
- **Obfuscation**
- **Licensing**

Encoding

Encoding converts PHP scripts from human-readable text files, to obscure binary files, which contain platform-independent Zend Intermediate Code. It enables you to distribute your application or your Web site's PHP scripts to end users working off any platform supported by the Encoder, without disclosing your source code. Business-oriented developers will find the Zend Encoder to be a mission-critical part of management strategy for protecting the intellectual property of their applications.

Encoded files can be read transparently by the Zend Optimizer™, which is available for free at <http://www.zend.com/store/>.

Parts of the Zend Optimizer are also embedded in the Zend Encoder, so encoded PHP scripts are optimized during the encoding process. This improves script performance and makes reverse engineering more difficult.

Obfuscation

Code Obfuscation lets Independent Software Vendors (ISVs), developers and businesses improve intellectual property protection from reverse engineering by obfuscating PHP source code.

Zend Guard enables the user to select the specific types of code elements to obfuscate; you can use Zend Guard to:

Remove:

- PHP DocBlocks
- Line Numbers

Obfuscate:

- Variables
- Functions
- Classes
- PHP Internal Functions

Note:

Code components (specific functions, function calls, concatenations, etc.,) that are added to the Exclude List will not be obfuscated.

Zend Guard supports the latest industry standard with full support for PHP 5.2x.

Licensing

Zend Guard's **Licensing feature** enable ISVs to manage the commercial distribution of their PHP applications by generating license keys and creating files that require a license key to operate. This protects code from copyright infringement. Software vendors can easily specify license models without changing their application's source code.

Applications will not run unless the proper software license is found.

Workspace Overview

Zend Guard's user interface is organized into 3 views (Guard Explorer, Project Editor and Monitoring). The Explorer is used to navigate through Zend Guard project files; the Project Editor is used to set and edit project configurations; the monitoring view displays editing issues while they occur and system messages when encoding projects.

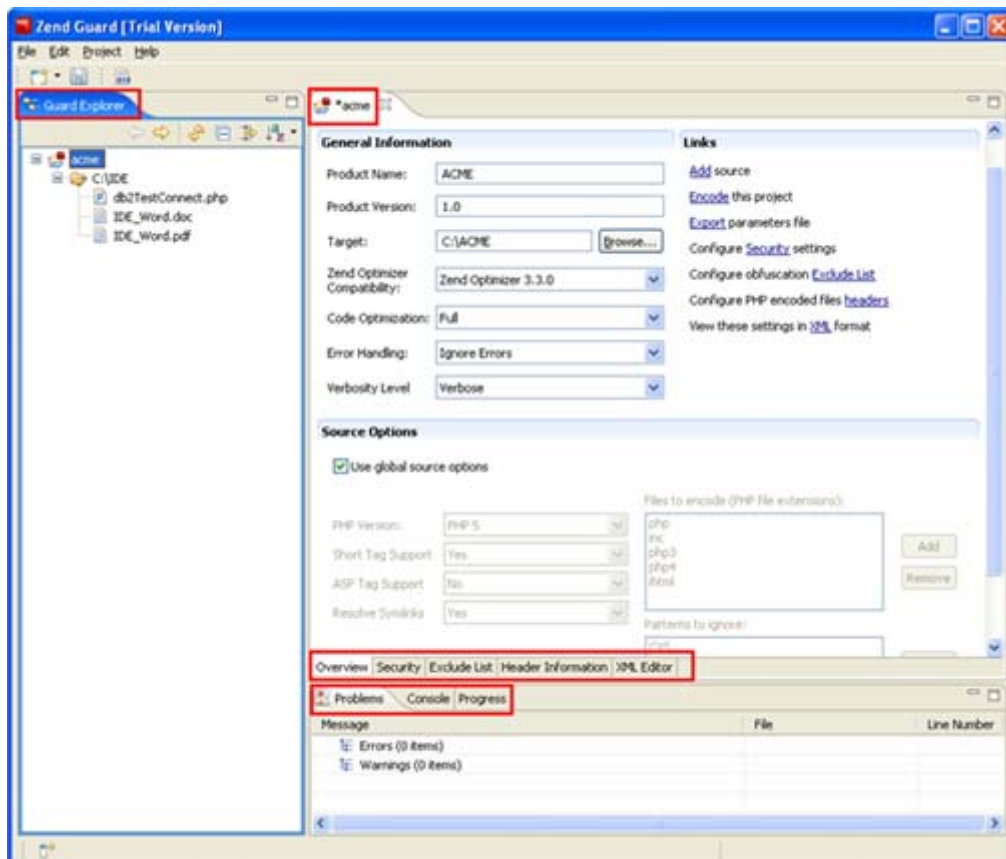


Figure 2 - Zend Guard GUI

Guard Explorer

This view is organized in the form of a browser tree. It contains all of your Zend Guard Projects, the projects' directories and the project files.

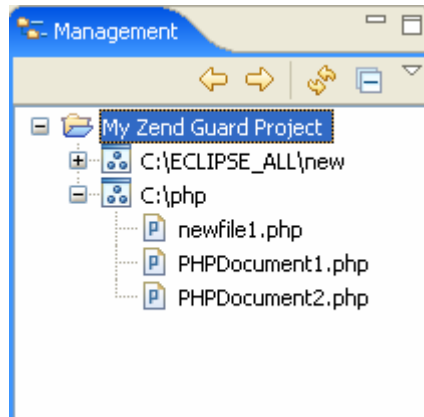


Figure 3 - Guard Explorer

Project Area

The Project view is used to manage Zend Guard projects from a technical standpoint. It is where the project is named, sources are added, encoding, obfuscation and security parameters are set, etc.

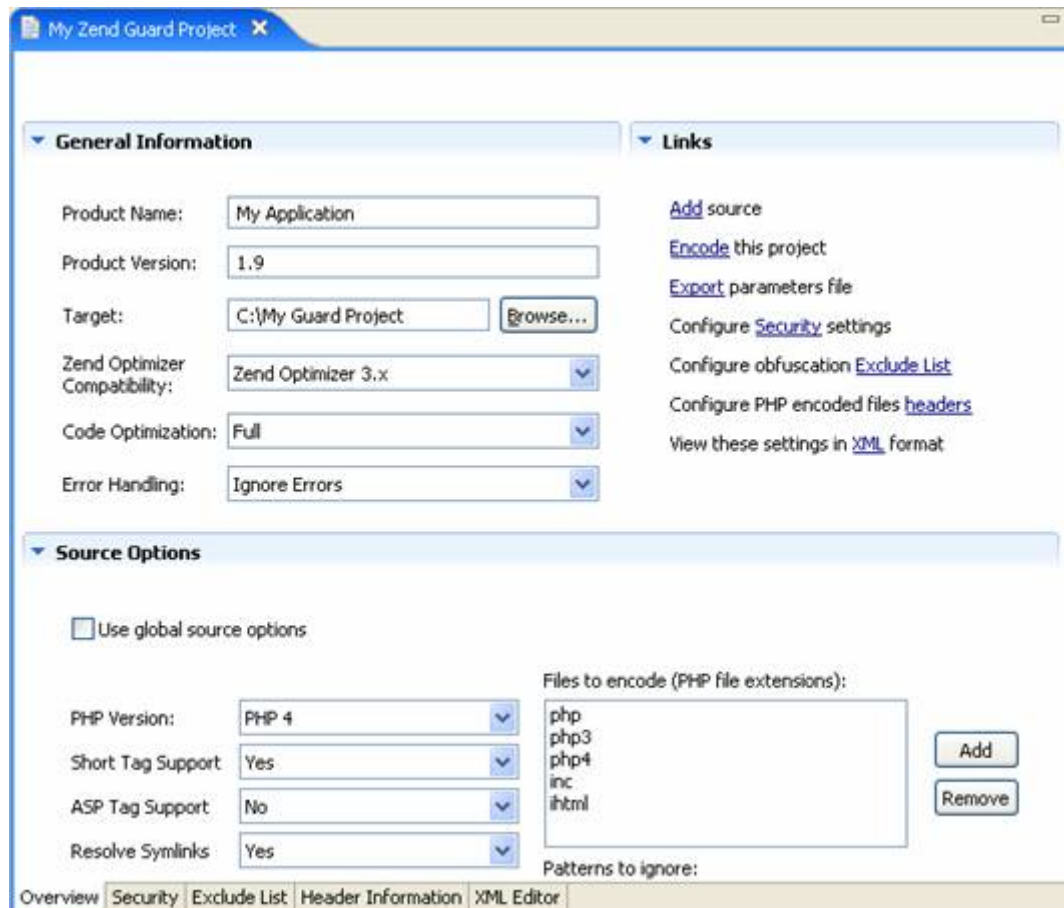


Figure 4 - Project View

The Project view includes the following five tabs:

- Overview - Includes General Information regarding the application (Product) that you are protecting via the Zend Guard as well as the Source Options for the Product (PHP version, tag support, symlinks, files to encode, patterns to ignore).
- Security - Used to set the options for Licensing, Encoding and Obfuscation.
- Exclude List - This is a list of code entities that should not be obfuscated. You can enter names manually and can also have Zend Guard generate a list of suggested entities that should not be obfuscated.
- Header Information - This enables you to insert meta information (e.g., copyright notice, version information, etc.) into your encoded application.

It also enables you to customize the message is displayed if the Zend Optimizer is not installed.

- XML Editor - The XML file contains all the information regarding the Zend Guard project. The Zend Guard engine uses this file as the instruction input; all obfuscation and related operations are based on the information contained in this file. You may edit the file manually.

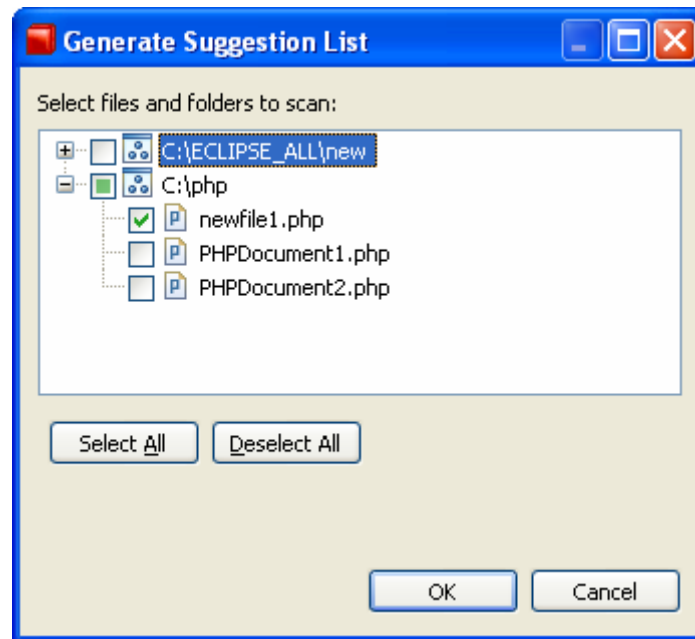


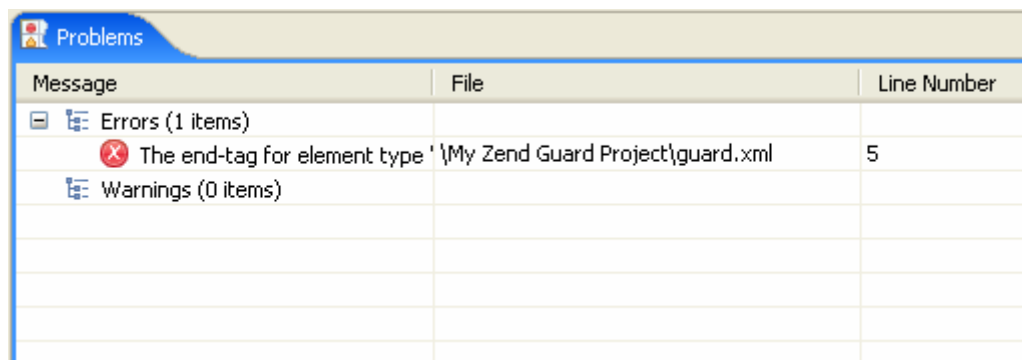
Figure 5 - Suggestion List

Monitoring Area

The Monitoring Area view lists all Errors and Warnings in the Zend Guard Project including an explanation of each, the specific file that generated each error/warning and the line numbers within the file that caused the error/warning.

The area contains three tags:

- **Console** - displays system messages when encoding projects such as encoding errors that occur in php files, the file name's description, error line number and a link to the Zend IDE (if the IDE has been integrated see "[Configuring Integration with Zend IDE](#)")
- **Problems** - displays XML validation issues in detail (including an explanation and the specific line that generated each error/warning).
- **Progress** - shows progress for jobs that are running.



Message	File	Line Number
Errors (1 items)		
The end-tag for element type ' 'My Zend Guard Project\guard.xml		5
Warnings (0 items)		

Figure 6 - Problem View

Menu Options









Zend Guard Main Menu

This table lists the Main Menu and short-cut key options. These include Project, Tools, and Help.

Menu	Options		Shortcut Key
File	New	Open the New Project wizard	Ctrl + N
	Save	Save currently opened Projects settings	Ctrl + S
	Save As	Save current settings as a new project (or overwrite an existing project that is not currently open)	
	Import	Imports a Zend Guard Project	
	Export	Exports a Zend Guard Project	
	Exit	Closes the Zend Guard without losing changes.	Alt + X
Edit	Preferences	Set default user interface and project settings.	
Project	Configure	Edit and set default project settings.	
	Variables	Map locations on the machine to the project.	
	Encode	Start Encoding	
Help	Help Contents	Opens the online help.	
	Register	Register the product.	
	About	View the current product's version details.	

Zend Guard Toolbar

This table describes the Toolbar and Explorer icons.

Toolbar Icon	Description
	New Project - Opens the New Project wizard.
	Save - Saves the current project.
	Encode - Encodes the project.
	Previous \ Next - Navigates between the previous and next files or folders to be encoded in your project.
	Refresh - Refreshes the current Views.
	Collapse - Collapses the Project tree.
	Filter - Displays the available filters.
	Sort - Sorts files by Name or Type.

Right-Click Menu Options

The Management view enables several context sensitive right-click menu options, listed below.

Projects

Place the cursor on a Zend Guard *project* and right click anywhere in the view to open the following menu.

Menu Item	Explanation
New*	Opens a Wizard to create a new Project or a new License.
Configure	Moves focus to the Project view .
Path Variables	Opens the "Configure Path Variables" dialog. Used for adding and editing project path variables.
Encode Project	Encodes the project and sends it to the target (output) folder.
Rename Project	Opens a dialog for changing the project's name.
Add Folder	Browse to add a file folder to the project.
Add Files	Browse to add individual files to the project.
Delete	Deletes the project's target files and optionally, the project's source files.
Close Project(s)	Closes the Zend Guard project(s).

Menu Item	Explanation
Import*	Imports a Zend Guard project (source files) or a project's parameter files.
Export*	Exports a Zend Guard project source files or the project's parameter files.
Refresh View*	Refreshes the project.

*Options available even when there are no Projects.

Folders

Place the cursor on a Zend Guard project *folder* and right click anywhere in the view to open the following menu.

Menu Item	Explanation
New	Opens a Wizard to create a new Project or a new License.
Configure	Moves focus to the Project view .
Ignore Resource (Don't Copy)	The resource will not be encoded or copied to the target location
Exclude Resource (Copy as is)	The resource will be copied to the target location as is (not encoded)
Include Resource (Encode & copy)	Appears only if one of the options "Ignore Resource" or "Exclude Resource" have been applied to the folder. Clicking "Include Resource", returns the folder to it's original state (i.e. it will be encoded and copied to the output location)
Import	Imports a Zend Guard project (source files) or a project's parameter files.
Export	Exports a Zend Guard project source files or the project's parameter files.
Edit Path	Select a file and choose the edit option. This opens the selected file for editing in Zend IDE (this option is only available for imported resources).
Remove Source*	Removes the folder from the project (this option is only available for imported resources).
Refresh View	Refreshes the project.

Files

Place the cursor on a Zend Guard project *file* and right click anywhere in the view to open the following menu.

Menu Item	Explanation
New	Opens a Wizard to create a new Project or a new License.
Configure	Moves focus to the Project view .
Open with Zend IDE	Opens the selected file in Zend IDE for editing and debugging code (available when the Zend IDE integration is active and configured).
Ignore Resource (Don't Copy)	The resource will not be encoded or copied to the target location
Exclude Resource (Copy as is)	The resource will be copied to the target location as is (not encoded)
Include Resource (Encode & copy)	Appears only if one of the options "Ignore Resource" or "Exclude Resource" have been applied to the file. Clicking "Include Resource", returns the file to it's original state (i.e. it will be encoded and copied to the output location)
Import	Imports a Zend Guard project (source files) or a project's parameter files.
Export	Exports a Zend Guard project source files or the project's parameter files.
Edit Path	Opens the open File dialog to edit the path to the file (this option is only available for imported resources).
Refresh	Checks all the project files and folders to see if anything was changed or deleted and updates the project files (similar to refreshing the project files list).

Working with Zend Guard

This section covers the following topics:

- **Getting Started** - How to create a project file.
- **Modify Project Contents** - How to add and remove files and folders from a project.
- **Basic Tutorial** - How to define encoding/obfuscation and security settings.
- **Preferences** - How to set Project preferences.
- **Overriding Project preferences** - see how to change default preferences for files and folders in the same project.
- **Integration** - How to configure the integration with the Zend IDE.
- **Import** - How to import Zend Guard 4 projects
- **Path Variable Preferences** - How to map file locations to handle several contents under the same project.
- **Editing Path Variables** - How to edit the path variables.

Getting Started

This procedure describes how to create a project for the first time. Projects are used to define specific settings for a group of PHP files that represent a complete PHP application. Projects are a collection of source inputs such as files and paths of files to be encoded. Inside a project you can define the specific encoding, obfuscation and license settings that will be applied to the files related to the project. There are also additional options for **excluding certain** entities.

There are three main steps to encoding PHP files:

1. Creating a Project
2. Configuring Encoding and Obfuscation settings
3. Executing encoding and obfuscation on the project



To create a new project

1. Select the "Create New Project" icon to open the "New Project Wizard."

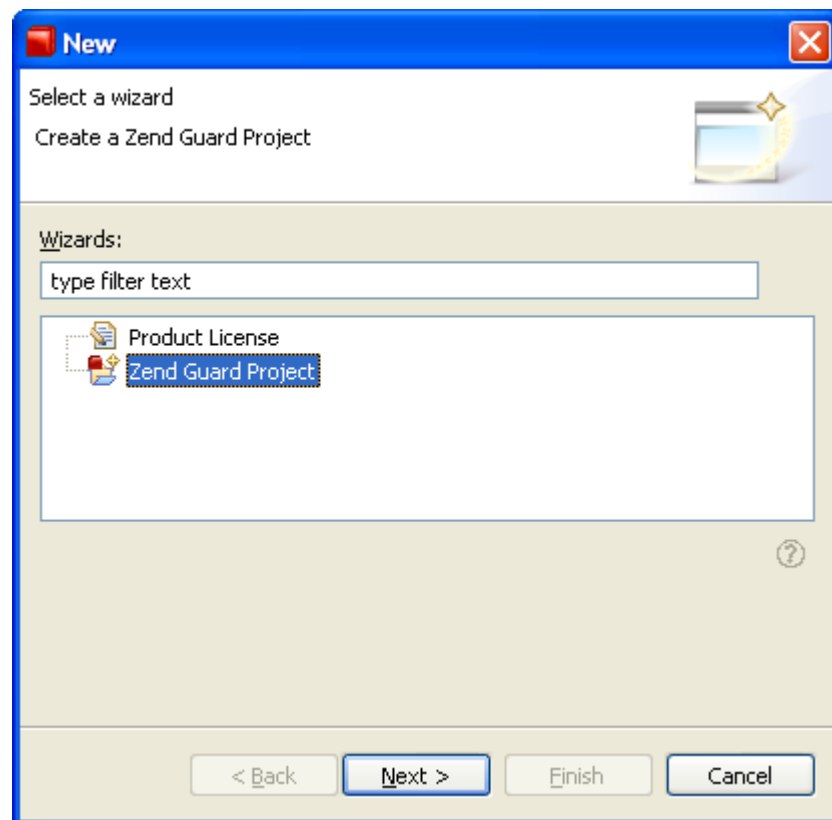


Figure 7 - Project & License Wizard

2. Click **Next** to define the project's basic settings including:
 - **Project**
 - **Name** - The project name. Make sure the project name does not

include spaces, use "_" as a separator if necessary.

- **Contents** - These are the files/folders that comprise the bulk of the Zend Guard project.
 - **Product**
- **Name** - The name of the (end) product.
- **Version** - The current version of the (end) product.
 - **Target directory** - The directory that will hold the (end) product files after creation by the Zend Guard. The target directory must not be the same as the source directory.

The screenshot shows a Windows-style dialog box titled "Zend Guard Project" with a close button (X) in the top right corner. Below the title bar, it says "Create Zend Guard Project" and features a folder icon. The dialog is divided into two main sections: "Project Information" and "Guard Product Information".

Project Information:

- Project name:** A text box containing "MY ZEND GUARD PROJECT".
- Project contents:** A checkbox labeled "Use default" is checked.
- Directory:** A text box containing "C:\Documents and Settings\Evan.ZENDNET\Zend\" followed by a "Browse..." button.

Guard Product Information:

- Product Name:** A dropdown menu showing "MY PRODUCT NAME".
- Product Version:** A dropdown menu showing "VERSION".
- Target Directory:** A dropdown menu showing "C:\TARGET_DIRECTORY\" followed by a "Browse..." button.

At the bottom of the dialog, there are four buttons: "< Back", "Next >", "Finish", and "Cancel". The "Next >" button is highlighted with a blue border.

Figure 8 - Initial Project Settings

3. Add the files and click **Next**. The Configure Source dialog will open. Enter the Source configuration settings.

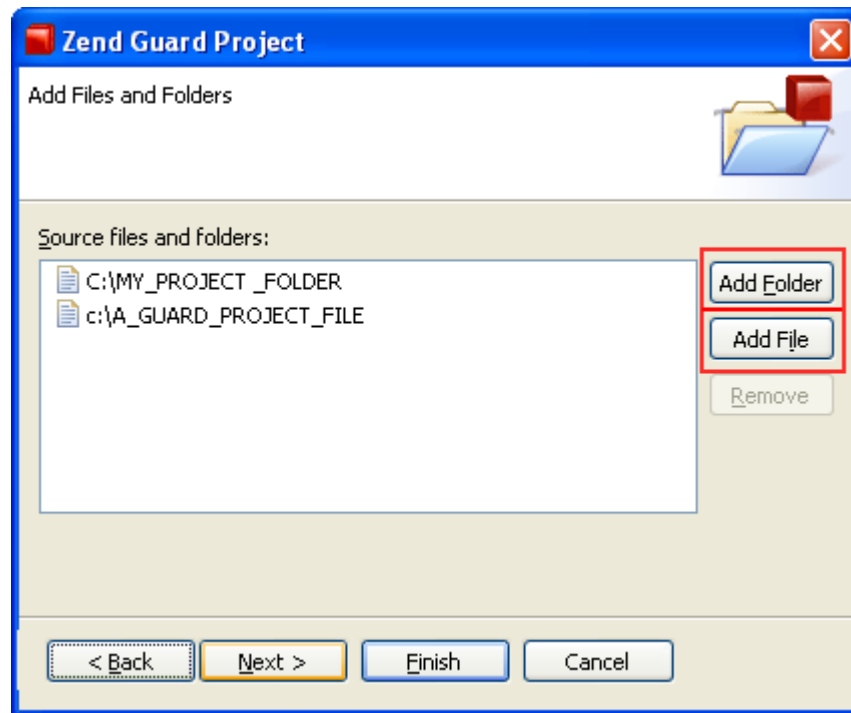


Figure 9 - Configure Source Dialog

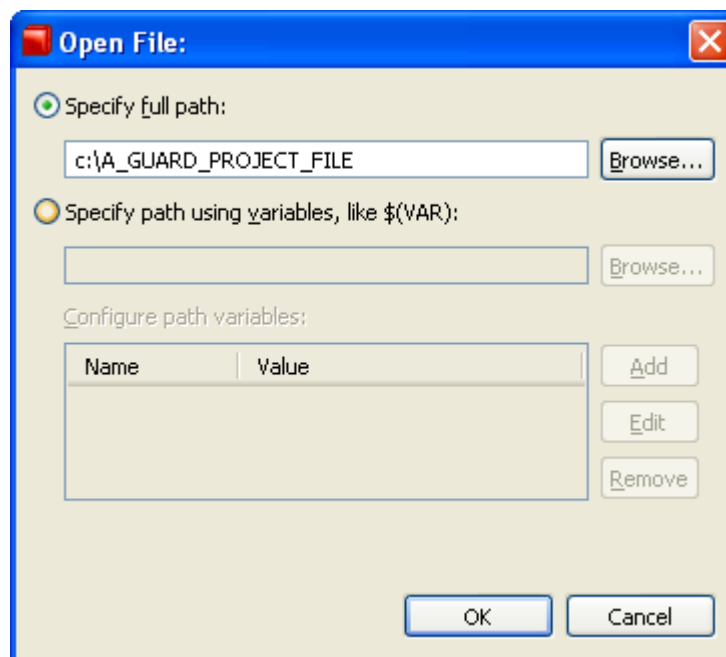
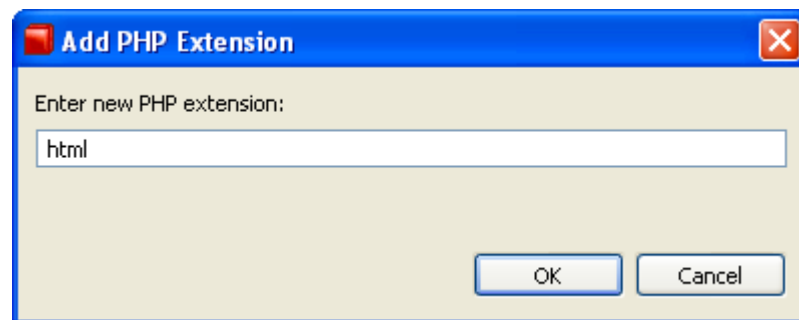
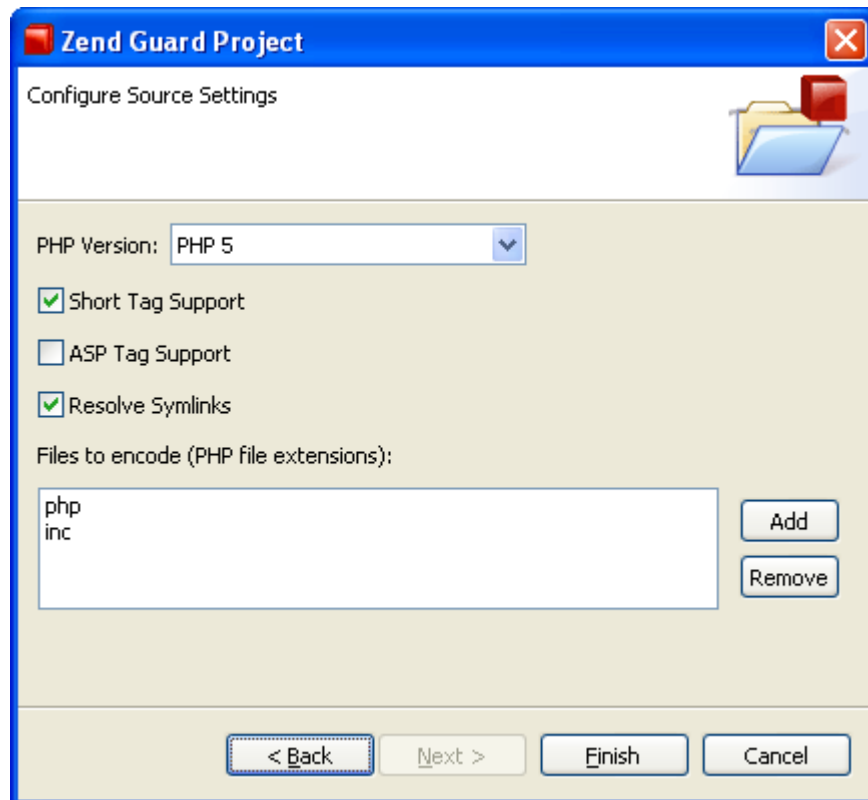



Figure 10 - Specify Source Path

4. In order to add an extension type (in this example - *.html), click Add, enter the extension and click **OK**.



5. Click **Finish** to create the project.
6. Click **File | Save** or click the Save icon  and the new project will be saved.

A new project will be added in the Guard Explorer section. Navigate through the different **tabs** to configure the project's specific settings.

Note:

To create evaluation software that expires after a designated period of time, use the [license-file expiration options](#) on the License tab.

Modifying Project Contents

The following procedures describe how to modify project contents.

There are four main options for modifying content:

- Adding a folder to a Project
- Adding selected files to a Project
- Removing files and folders from a Project
- Ignoring and Excluding Resources

Before adding or removing files and folders it is important to know that:

- Single files or entire folders can to be encoded.
- **Folder Structures** are treated recursively - any files found in the immediate folder or below, which match the **Extensions to Encode Definitions** will be processed for encoding.
- The **Copy Non-PHP Files option** - copies files whose extensions are not specified in **Tools | General Settings**, to the output directory. You should evaluate the contents of the structures before you select **Copy Non-PHP Files** as an encoding setting.
- **Defined File Types** are files that match the extensions list as set in **Tools | General Settings**. By default, these are limited to: **php, phtml, inc, php3, php4 and php5**.



To add a folder to a project:

1. In the Guard Explorer, open the right-click menu and select **Add Folder**.
The folder browser will open.
2. Select the folder to add, and click **OK**.

The new folder will be added into the project and will be visible in the Guard Explorer.



To add selected files to a project

1. In the Guard Explorer, open the right-click menu and select **Add Files**. The file browser will open.
2. Select the file to add, and click **OK**. (For multiple files use CTRL while selecting the files.)

The new files will be added into the project and will be visible in the Guard Explorer.

**To remove files or folders from a project:**

1. Select a file or folder in the Guard Explorer..
2. Open the right-click menu and select **Remove from project**. The selected file or folder will be removed from the project (For multiple files use CTRL while selecting the files.)

The removed files and folder will be removed from the project and will no longer be visible in the Guard Explorer or in the project's directory.

Note:

Some applications are compatible with both PHP 4 and PHP 5. Encode them using the PHP version used on the server.

If you want to keep resources in the project directory but you don't want them to be sent to the output location, use the "Ignore Resource" option.

**To Ignore resources:**

1. Select a file or folder in the Guard Explorer.
2. Open the right-click menu and select "**Ignore resource**".
3. The "ignored" indicator will appear on the resource.

The selected file or folder will be ignored (i.e. not encoded and not copied to the output location).

If you want to have resources in the output location that were not encoded, use the "Exclude Resource" option.

**To Exclude resources:**

1. Select a file or folder in the Guard Explorer.
2. Open the right-click menu and select "**Exclude Resource**".
3. The "excluded" indicator will appear on the resource.

The selected file or folder will be excluded (i.e. not encoded and copied to the output location).

Basic Tutorial

Once you have decided which PHP files need to be encoded and the **Obfuscation type** you want to apply to the code. **Create a Zend Guard project** and **configure the project's settings** according to the following questions:

- Is the code intended for mass deployment?
- How important is the code (i.e. is it expensive intellectual property)?

If both the answers are "yes", it is better to use all Obfuscation types to secure more code elements.

Encode Only

To encode without obfuscation, do not select any Obfuscation options.

List all code entities that will not be encoded in the **Exclude List**.

Note:

Encoded files which have obfuscated *local* variables only are compatible with Zend Optimizer version **2.6** and above.

Encoded files which have obfuscated variables and also have obfuscated **functions, classes** and/or **PHP Internal symbols** require the **latest version** of Zend Optimizer (PHP4 Optimizer 3.0 and above; PHP5 Optimizer 3.3 and above).

The Zend Optimizer is available free of charge, from:
<http://www.zend.com/downloads>.

Scan Feature

The Scan feature (on the Exclude Tab) scans the code in order to locate entities that should be added to the Exclude list. Among other things, it identifies strings and functions with the same name. You can discard any suggestions. However, the following situations require adding entities to the Exclude list manually.

Situations Requiring Adding Code Entities to the Exclude List

Testing/Debugging Process	<p>Entities that prevent the application from working properly, (after obfuscation).</p> <p>These entities will generate "Function not defined", "Method not defined" and "Class not defined" message types and will list the code file name and specific line. This can be useful for tracking problems as the obfuscated name may be meaningless. Having</p>
----------------------------------	--

	the file name and line in code helps identify which function/class/method has been called.
Functions located during debugging	Functions located during the debugging/testing
Function not defined	Functions that generated "Function not defined" and "Class not defined" message types that appeared only after obfuscation.
Concatenating Strings into Function Names	The only instance the Suggest function cannot identify is when Concatenating Strings into Function Names. This includes instances where the code calls an indirect function name and not the function's real name. This occurs when the real function names are not identified in the code as functions, but rather as strings. The Suggest feature searches only for functions in the code.
Indirect Functions	User functions that are used indirectly or are called from un-obfuscated script.
Indirect function calls	This occurs when referencing function calls through a variable holding the function name.
Functions defined in un-obfuscated code	This includes functions that cannot be automatically identified through the setup process. These include indirect functions and concatenated functions .

Note:

Errors that occur in the code **before** obfuscating indicate a problem in the actual code.

Exclude List

The **Exclude List** contains all entities will **not** be obfuscated. Use the Scan feature to recommend functions to be added to the Exclude list.


The Scan feature automatically identifies most functions that should not be obfuscated. Run the Scan feature to locate functions that should not be obfuscated.

Secure the Project

Once all the settings have been configured the project can be encoded.



To Encode a Project

1. Make sure all the Project settings are configured to your requirements.
2. In the main toolbar click  (Project | Encode).

-Or-

In the Overview Tab click the "Encode Project" link.

The secured files will be placed in the Output Location defined in the Overview Tab.

Preferences

The preferences menu is a container for setting and viewing Zend Guard project settings.

The preferences is accessed from the edit menu by clicking `Edit | Preferences`.

Note:

Preferences can also be configured for specific files/folders belonging to a project. See "[Overriding Project Preferences](#)".

The preferences menu contains configuration options divided into the following categories:

- **Editors** - Defines the XML editor's behavior when switching between elements. The options determine what to do with changes made to XML file content.
- **Encoding and Obfuscation** - set default preferences for guarding project content.
Remove: PHPDoc Blocks and Line Numbers. Obfuscate: Variables, Functions, Classes and PHP Internal Entities.
- **Header Information** - append code and information to the beginning of each encoded file. This allows you to insert meta information (e.g., copyright, version, etc.) into your encoded application. It also allows you to customize the message that is displayed if the Zend Optimizer is not installed. See the "[Header Information Tab](#)" for complete information.
- **License Keys** - Use this to generate new License Keys and to detach old licenses. Also, see [Creating a License](#) and [License Files](#).
- **Source Options** - Define file and version specific handling options (see table below for a description of each option).
- **Zend IDE** - Use this to set the path to Zend IDE. When enabled, you can open PHP files from Zend Guard and edit them in your Zend IDE.

Preferences are applied to all projects unless **specified** and changes to the preferences are applied to new projects (Existing Projects are not affected by changes done in the general preferences menu).

Source Options

Source Option	Description
PHP Version	PHP 4 or PHP 5.
Short Tag Support	Enable recognition of short PHP tags. Recognizes <code><?</code> as a valid PHP start tag. When this option is not selected, Zend Guard will not encode short tags; they will be treated as regular HTML.
ASP Tag Support	Enable recognition of ASP tags. Recognizes <code><%</code> as a valid PHP start tag. When not selected, code within ASP tags is treated as regular HTML.
Resolve Symlinks	Resolves Symbolic Links before encoding (not applicable in Windows). A symbolic link (often shortened to symlink and also known as a soft link) consists of a special type of file that serves as a reference to another file or directory. Unix-like operating systems in particular often feature symbolic links.
Files to Encode	Lists the file extensions for Guard to encode (extensions not listed will not be encoded). File extensions that are not listed here and in "Patterns to Ignore" will be sent as-is to the output folder.
Patterns to Ignore	Files matching these patterns will not be encoded when encoding a directory, nor will they be copied as-is to the target directory. By default, the list contains the CVS directory and <i>cvsignore</i> files (includes Wildcards '*').

Non-PHP CODE Encoding sends output of the PHP encoded files to the target directory by default. However, some files will not be encoded (e.g., images that need to be copied to the output directories). *non-PHP project files will be sent as-is to the* output directory automatically if they are not included in the list of Patterns to ignore.

Overriding Project Preferences

This procedure describes how to override the default preferences that are set in:

Edit | Preferences.

Override preferences is only done to existing projects.

The override feature is used when configurations different than the ones defined in the Project should be applied to a resource (file or folder). One of the benefits of this option is when a certain pattern is selected that includes a file name. For example; if you choose to exclude the pattern test*.php and you have a file called contest.php the file will also be excluded. Applying the override option to a selected file will remove the association with the exclude settings.

Preferences can be changed per folder or single file. This option is useful for cases where a certain file or folder contains different characteristics such as situations when an entire project consists of PHP5 code and only one folder contains code written in PHP4.



To Override Project Preferences

1. Open the Project in the Guard Explorer
2. Click on a File or Folder
A blank Overview Tab will be displayed with a single option:
"Override Project Configuration".
3. Click this option.
Configuration Options will be enabled.
4. Set the configuration options for the selected file or folder
5. To save the changes, go to the main toolbar and click Save.

Repeat this process for each file and folder that required unique settings (different than the **default Project preferences**).

Configuring the Integration with Zend IDE

The following procedure describes how to configure the integration with Zend IDE. The integration enables users to extend Zend Guard with the rich editing features included in the Zend IDE. This provides users with a seamless workflow for fixing and modifying code through Zend Guard. This enables you to open files and fix errors in Zend IDE, seamlessly.

Before configuring the integration make sure you have both Zend Guard and Zend IDE installed on the same machine. It is also useful to already know the location of the Zend IDE's installation directory before setting-up the integration.

Settings have to be configured in both applications in order to successfully start the integration between Zend IDE and Zend Guard.



To configure the Zend IDE settings:

1. Make sure the Zend Guard is installed on the same machine.
2. In **Zend IDE** go to: **Tools | Preferences** and select the **Desktop tab**.
3. Go to the Desktop tab's **General Settings** section and enter the path directing to the Zend Guard program file in the Zend Guard Path field.



To configure the Zend Guard settings:

1. Make sure that Studio is installed on the same machine.
2. Go to: **Edit | Preferences | Zend IDE**.
3. Enter the path to Zend IDE and click **Apply/OK**.

Once the configuration has been completed, a new option will be added to the right-click menu in the Guard Explorer called "Open with Zend IDE". Clicking this option will automatically open the file for editing in the Zend IDE.

Importing an Existing Project

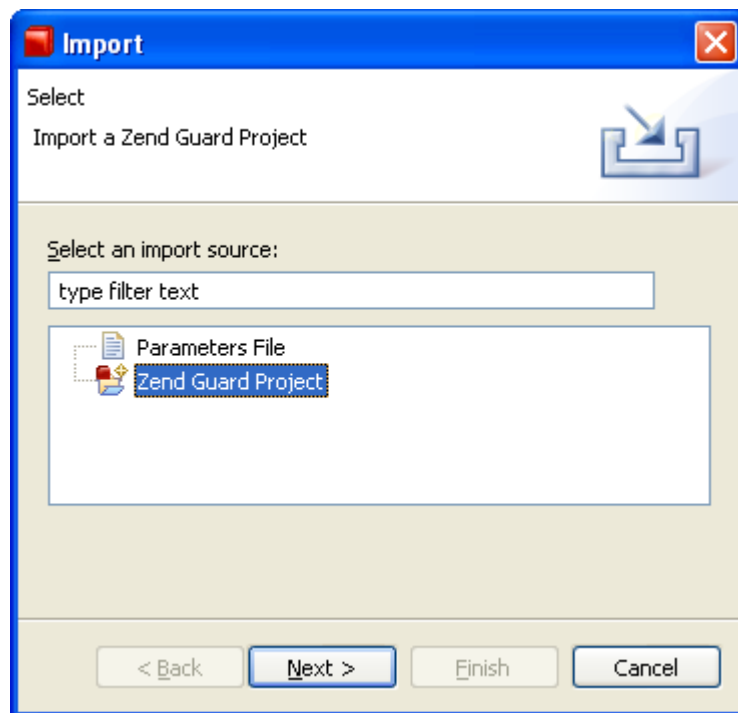
This procedure describes how to import a project that was made with Zend Guard version 4 and earlier. Using this procedure will ensure seamless import of your existing project settings and configurations to the Zend Guard Eclipse based environment.

Only use this procedure if you have upgraded Zend Guard from a previous version.



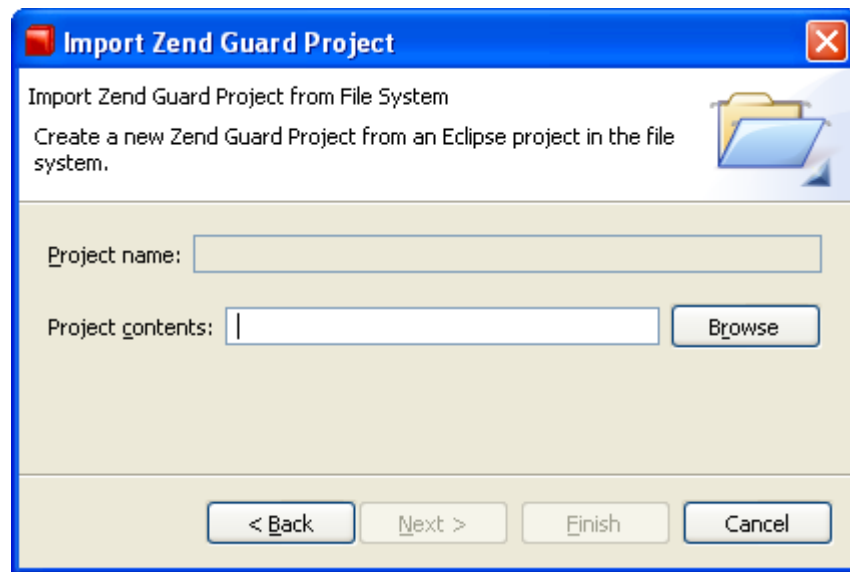
To Import an existing project:

1. Go to the Management view. Right click and select **Import**
-or- In the Guard Explorer right-click and select **Import**.
The Import dialog will open.




Import Project

2. Select **Zend Guard Project** and click **Next**. Use the Browser to locate an existing project or to locate an Eclipse project from the file system.



Project Contents (Imported)

3. Browse to find the project to be imported and click Next.
4. Click **File | Save** or click the Save icon  and the project will be saved.

The imported project will now be ready for use.

Path Variable Preferences

Path Variables enable to map locations on your machine in order to easily handle content located in different directories under the same project. This option is only enabled (active) when the topmost node is selected in the Guard Explorer.

A path variable is a name that is mapped to a specific location on the machine. By using a path variable, you can share projects containing linked resources with team members without requiring the same directory structure as on your file system.

Include path variables can also be added to a project's include path.

The Path Variables menu is accessed from Project | Path Variables.

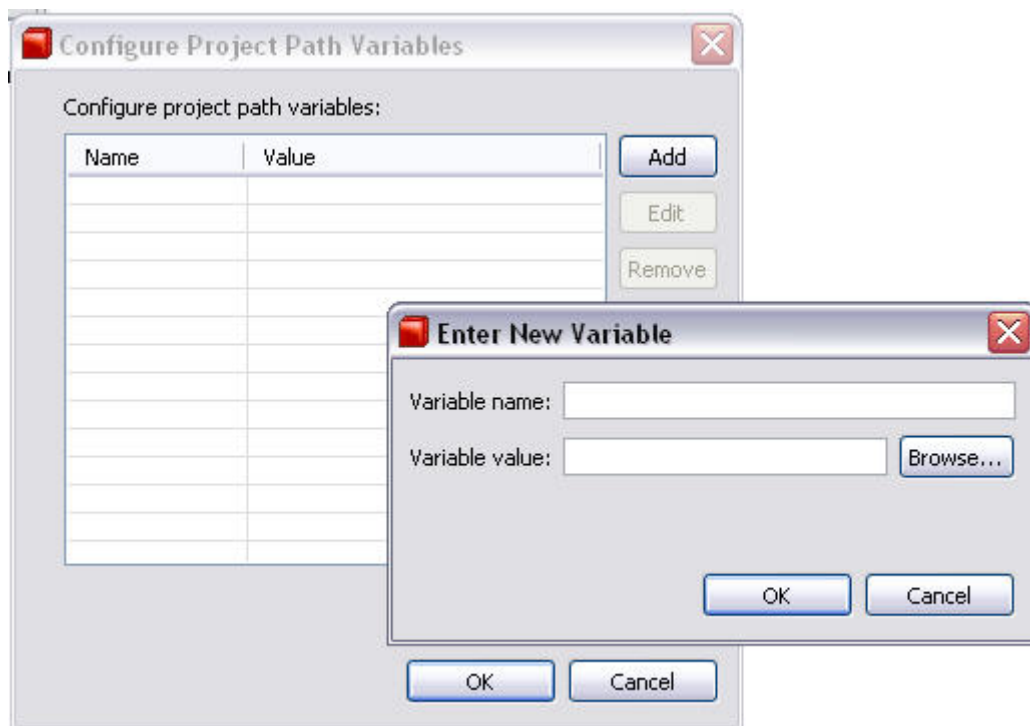


Figure 11 - Path Variables Menu

The Path Variables menu allows users to **add edit and remove path variables**.

Editing Path Variables

This procedure describes how to Use Path Variables in a Project Context

To enable the Path Variables menu make sure the topmost node is selected in the Guard Explorer.



To do add a new Path Variable

1. Go to Project | Path Variables
2. Click Add.
A new dialog called "Enter New Variable will be displayed.
3. Enter the Variable Name - Select a name that will be meaningful to the Projects context.
Note: variable names should not include empty spaces and slashes, use underscores or uppercase characters to separate between names.
4. Use the Browse button to locate the folder.
5. Click OK to save and close the variable.

A new variable will be added to the list of the project's available variables.
Once the variable has been added the new content can be added to the Project.



To add new content to the project:

1. Go to the Guard Explorer and select the Project folder (the topmost node in the tree).
2. Right Click on the project folder to open a menu.
3. Select Add File or Folder (depending on the content that you want to add).
The "Open File/Folder" Dialog is displayed.
4. Select the option "Specify path using variables, like \$(var)".
5. Select the variable from the list and click OK to save and close.

The new File/Folder will be added to the Guard Explorer

Project Variables can also be edited and removed from the project settings. Editing and removing is only enabled after a Project Variable has been added to the project.

**To Edit/Remove Path Variables from a Project:**

1. Select the topmost node in the Guard Explorer.
2. Go to Project | Path Variables.
3. Select a variable from the list and Click Edit to open the Edit dialog and modify the variables name -or- remove to delete the variable from the project.

After clicking OK the changes will be immediately applied to the Guard Explorer.

Note:

Changing a Variable name to a name that does not exist will remove the file/folder from the Guard Explorer, renaming the file back to the correct name will restore the file/folder to the explorer.

Also, removing a variable from the variables list will also delete the content from the project.

Tabs

The tabs section is the main area for viewing and configuring advanced settings for projects files and folders.

Note:

If the options in the tabs are disabled the selected project and its included files and folders will inherit the **default preferences**. To override these preferences see "**Overriding Project Preferences**".

Tabs are used to define and configure the following:

- **Overview** - a general overview of your Zend Guard projects and access to most other available functionality.
- **Security** - **License** and **Security** configuration options including encoding and obfuscation.
- **Exclude** - lists entities that should **not** be obfuscated.
- **Header** - to **append code and information** to the beginning of each encoded file.
- **XML** - displays the entire XML configuration file for editing.

Overview Tab

The Overview tab provides a general overview of your Zend Guard projects and access to most other available functionality.

This tab is divided into two sections as follows:

- **General Options** - General configuration Settings
- **Source Options** - PHP code global and local settings

General Options

Zend Guard's **Overview** tab sets the application's general configuration and default project settings

The encoding settings determine the type of encoding to apply to your code and offer other security and obfuscation options to further enhance the safety precautions used to protect your code.

Option	Description
Product Name	Set the name for the Product.
Product Version	Set the Product version.
Output Location	The folder in which the encoded files will be placed.
Zend Optimizer Compatibility	<p>PHP 4 project files that have been encoded or whose local variables (only) are obfuscated are compatible with Zend Optimizer 2.6.</p> <p>PHP 5 project files that have been encoded or obfuscated more strongly, require Zend Optimizer 3.3 (or greater) in order to run.</p> <p>In general, Zend Optimizer 3.3 requires PHP 4.3 and higher.</p>
Code Optimization	<p>Optimization enables faster execution and reduces the CPU load for the server. Some scripts may not support optimization (generally, scripts that interface with COM or Java objects). If you experience problems with the encoded scripts, try reducing the optimization level.</p> <p>None - Disables code optimization.</p> <p>Minimal - Use if application does not perform correctly.</p> <p>Full - Recommended optimization level. Optimized files run faster and are more difficult to attack.</p>

Error Handling

Ignore Errors - Encoding **will not** be terminated when errors occur in the PHP code.

Exit on Error - Encoding will stop and display the errors in the Zend Guard's messages area.

Note:

When integrated with Zend IDE, users can investigate errors with Zend IDE directly from the messages area. To do so, click on a message.

Verbosity Level

The options are:

Verbose - Provides detailed information.

Silent - Lists Errors only

Links

Click the linked option to open the corresponding Guard dialog:

- Encode Project - a shortcut to start encoding the project.
- Add Source - Opens a dialog for adding additional source files to the project.
- Export Parameters File - Opens a dialog for generating an XML file containing the project's encoding configurations for "[Encoding External code using the XML File](#)".
- Configure Security Settings - Jump to the **Security** tab.
- Configure Obfuscation Exclude List - Jump to the **Exclude** tab.
- Configure PHP Encoded File Headers - jump to the **Header Information** tab.
- View these settings in XML Format - Jumps to the **XML Editor** to display the settings in XML format.

Source Options

Set the following Global and Local Options.

Option	Description
Use Global Source Options	Check this to use the Global options. When this is not checked, the settings will apply only to the specific files or folders selected by the management tab.
PHP Version	Set to encode either PHP 4 code or PHP 5 code.
Short Tag Support	Enable recognition of short PHP tags. Recognizes <code><?</code> as a valid PHP start tag. When this option is not selected, Zend Guard will not encode short tags; they will be treated as regular HTML.
ASP Tag Support	Enable recognition of ASP tags. Recognizes <code><%</code> as a valid PHP start tag. When not selected, code within ASP tags is treated as regular HTML.
Resolve Symlinks	Resolves Symbolic Links before encoding (not applicable in Windows). A symbolic link (often shortened to symlink and also known as a soft link) consists of a special type of file that serves as a reference to another file or directory. Unix-like operating systems in particular often feature symbolic links.
Files to Encode	Lists the file extensions for Guard to encode (extensions not listed will not be encoded). File extensions that are not listed here and in "Patterns to Ignore" will be sent as-is to the output folder.
Patterns to Ignore	Files matching these patterns will not be encoded when encoding a directory, nor will they be copied as-is to the target directory. By default, the list contains the CVS directory and <i>cvsignore</i> files.

Security Tab

This tab contains **License** and **Security** configuration options.

- **Licenses** - Licenses allow you to control the use of your encoded files as well as preventing piracy, unauthorized or invalid use. The Zend Licensing options enable you to configure **License options**, **Restrictions** and to create **License files**.
- **Encoding and obfuscation** - Encoding converts PHP files into encoded binary files. Obfuscation converts user generated names (of variables, functions, classes, etc.) into machine generated names that contain no application context. The Encoding and Obfuscation Tab is used to manage encoding and obfuscation options and configurations for the entire Zend Guard project. Individual options can be applied to all Guard files or to individual files, as required.

Note:

Once the files have been encoded they are no longer editable. Any attempt to modify the files once encoded will result in rendering the files unusable. This provides an additional security layer for protecting intellectual property.

Name	Description
Licenses	
Work Only with Encoded Files	<p>Creates encoded files that work only in conjunction with other encoded files created by the <u>same version</u> of Zend Guard.</p> <p>Enables you to encode your files so that they will work only with encoded files containing your signature and makes it more difficult for users to reverse-engineer your code using PHP's code introspection functions, and <code>include()</code>.</p> <p>This feature prevents users from attaching their own encoded file or using PHP's introspection functions to reveal information about the structure of the file</p> <p>If not selected the above protections are not enabled.</p>

Name	Description
Licensing	<p>Enables you to encode files to work <i>only</i> with the License files.</p> <p><u>Disabled:</u> Disables licensing support altogether.</p> <p><u>Enable License Support:</u> A valid license is required for the encoded files to load properly. The license itself is generated by clicking the link: "Generate Product License File" located on the Zend Guard's Security tab (or by clicking File New Product License). The license requirement is automatically enforced at all times.</p> <p>License Key</p> <p>Generated - Works with a license key generated by Zend Guard.</p> <p>External - Works with a Zend Guard generated license key saved in an External File.</p>
Expiration	<p>Determines expiration behavior for the encoded files. Can be used to limit a file's life span (e.g., sample version that will expire on selected date).</p> <p>This feature is not related to the License Manager and should not be confused with the expiration date of licenses.</p> <p>Encoded code will never expire: Disables the expiration date for encoded files.</p> <p>Encoded code will expire on: Sets the expiration date for encoded files. The encoded files will expire on the expiration date and be made unusable. Use this option to create for time-limited software, beta versions or releases that will expire on a specific date. This can encourage your customers to upgrade.</p> <p>Encoded code will expire after: Sets the number of years/months/days that the encoded file will work (starting from date the application was first used).</p>
<p>Note:</p> <p>To create evaluation software (Beta versions) that expires after a designated period of use, use the <i>Expiration</i> options.</p>	

Encoding and Obfuscation Methods

Remove:

- PHP DocBlocs - Removes PHP DocBlock sections while encoding.
- Line Numbers - Removes line numbering while encoding. Removing the line numbers when an error appears means that there will not be a line number to identify the location of the code that generated the error. This is an additional security precaution against reverse engineering.

Obfuscate:

Encoding is automatic. It converts PHP files into encoded binary files but does not obfuscate code.

- Variables - Scrambles the context by encrypting user generated variables.
- Functions - Scrambles the context by encrypting user generated function names.
- Classes - Scrambles the context by encrypting user generated class names.
- Apply selections to PHP built-in symbols -Scrambles the context by encrypting PHP internal symbols.

Exclude Tab

This list is a placeholder for entities that should **not** be obfuscated. In general these are **user functions**, class names, and class methods that are used indirectly or need to be called from an un-obfuscated script.

Entities can be added to the exclude list manually (click the **New** button) and by using the **Scan** option.

The main reason for adding entities to the exclude list is to facilitate the debugging, QA process. Some entities can cause code to fail if they are obfuscated. To prevent this from occurring in the final product version always test your code after obfuscating. Any entities that cannot be obfuscated should be added to the Exclude List.

This procedure describes how to use the Exclude List to define which code entities should not be obfuscated.

The Exclude List includes the following options:

- **Scan Code**
- **Add Entities to the List**
- **Import and export List contents**
- **Remove contents from the List**

To read more about what to exclude go to "**Excluding Functions**".

Note:

Errors that occur in the code before obfuscating indicate a problem in the code.

Scanning Code

The Scan feature scans the code in order to locate entities that should be added to the Exclude list. It identifies strings and functions with the same name. You can discard any suggestions.



To Automatically Scan Code

1. Click the [Scan](#) link. The code will be scanned and the suggestions displayed in a suggestions list.
2. Check the suggestions to be used, the others will be discarded automatically.

After the suggestions have been applied and code has been obfuscated, the code should be tested normally.

Adding Entities Manually

You can add entities to the Exclude List manually. Listed entities will ***not*** be obfuscated. In general, if an entity causes an error during testing add it to the exclude list to determine if the obfuscation caused the error.

Always Exclude

- **Indirect function calls.** These occur when referencing function calls through a variable holding the function name.
- **Functions Defined in Un-obfuscated Code**
 - Functions that cannot be automatically identified through the setup process. These include indirect and concatenated functions.
 - Functions located during the debugging/testing stage of the application.
 - Functions that generated "Function not defined" and "Class not defined" message types that only appeared after obfuscating the code.



To manually add entities:

1. In the Exclude List Tab, click New.
The Add Excluded Entities dialog will be displayed.
2. Enter the name of the entity, the wild card (*) can be used only at the end of strings when adding names manually to the exclude list (correct use: " test* "; incorrect use: " *test"). Click OK to save and close.
The entity will be added to the Exclude List.
3. -Optional- In the Exclude list's Reason column, Click to begin writing and describe why the entity was added to the Exclude List.

Once added to the list and as long as the check-box next to the name is full, the Entity will not be obfuscated.

Note:

Use the override feature when a certain pattern is selected that includes a file name. For example; if you choose to exclude the pattern test*.php and you have a file called contest.php the file will also be excluded. Applying the override option to a selected file will remove the association with the exclude settings.

Importing and Exporting Exclude Lists

This procedure describes how to import and export Exclude List settings. These options enable you to share your selection with other Zend Guard users.

Importing an Exclude List is based on the assumption that you already have an Exclude List file generated from Zend Guard.



To import an Exclude List:

1. In the Exclude List tab, click Import.
A browse dialog will be displayed.
2. Browse to locate the file and click Open.

Once added to the list and as long as the check-box next to the name is full, the Entity will not be obfuscated.

Exporting an Exclude List is the process of generating an external file containing all the entities and their descriptions (reasons) to an external file that can be reused with other projects and Zend Guard applications.



To export an Exclude List:

1. In the Exclude List tab, click Export.
A browse dialog will be displayed.
2. Browse to locate the place to save the file, name the file and click Save.
A new file containing the contents of the Exclude List and its details will be created in the specified location.

Once exported the file can be reused in other projects and Zend Guard applications to automatically populate an Exclude list.

Note:

The "Load from file" option does not validate the contents of the file; Zend Guard assumes that each line is a separate function. Zend Guard does not verify that the entities listed in the file are proper functions, etc.

Removing Content from the Exclude List

The procedure describes the different options for removing content from the Exclude List.

- Remove - will remove a selected entity from the Exclude List
- Remove Suggested - will remove all the entities that were added to the list as a result of the Scam process
- Remove all - will empty the list of all its contents.

Excluding PHP Entities

When using the Exclude List (in the Project Areas Exclude List Tab) there are several code related issues that should be considered.

The following describes the instances where it is recommended to add different entities to the Exclude List and how to optimize your selections.

Contents:

Functions Referenced via a Variable

Functions Passed via Arguments

Functions Implementing External Interfaces

Functions Used as Object Callbacks

Classes

Autoloading Classes

Exclude Application APIs

Functions Referenced via a Variable

Functions that are referenced by a variable that holds their name, should be added to the **Exclude List**.

```
function do_mysql_query($query) { ... }
function do_sqlite_query($query) { ... }

if($db == "mysql")
{
$query_function = "do_mysql_query"
}
else
{
$query_function = "do_sqlite_query";
}
$result = $query_function ("SELECT * FROM TABLE");
```

The functions *do_mysql_query* and *do_sqlite_query* should be added to the exclude list so their names will stay intact.

Functions Passed via Arguments

Functions that their name is passed to other functions through arguments (callbacks), should be added to the Exclude List. In the code example below, the functions *myerror* and *myfunc* are callback functions and should be added to the Exclude List.

```
function myerror() { ... }
set_error_handler('myerror');
- or -
function myfunc($data) { ... }
array_walk($array, 'myfunc');
```

Functions Implementing External Interfaces

Functions that implement an external interface (in this example: *rewind*, *valid*, *current*, *next* and *key*) should be added to the Exclude List otherwise the *c_iter* will no longer implement the *iterator* interface.

```
class c_iter implements Iterator {
function rewind() { ... }
function valid() { ... }
function current() { ... }
function next() { ... }
function key() { ... }
}
```

Functions Used as Object Callbacks

Functions and classes that are related to object callbacks should be added to the Exclude List.

```
class VariableStream {
    function stream_open(...) {}
    function stream_read($count) {}
    ...
}
stream_wrapper_register("var", "VariableStream");
```

In this example, the class name *VariableStream* and its methods (i.e. *stream_open*, *stream_read*) must be added to the Exclude List. (See <http://www.php.net/manual/en/function.stream-wrapper-register.php> for a complete list of callback names, classes and other such functions).

Classes

When the code refers to class names (or methods) through strings, the class name (or method name) must stay the same. Therefore the class name must be added to the exclude list (to avoid obfuscation).

Example:

```
<?php
```



```

class MyClass
{
    public function printName()
    {
        echo "John";
    }
    public function printLastName()
    {
        echo "Doe";
    }
}

$className = "MyClass";
// runtime error: after obfuscation MyClass is no longer the
class name
$obj = new $className();
$obj->printName();

// runtime error: after obfuscation MyClass is no longer the
class name
$clazz = new ReflectionClass("MyClass");
$obj2 = $clazz->newInstance();
// runtime error: after obfuscation printLastName is no longer
the method name
$method = $clazz->getMethod("printLastName");
$method->invoke($obj2);
?>

```

Autoloading Classes

Autoloading classes will not work since the filename on the disk would not match the obfuscated class name. The classes that are loaded through *autoloading* must be added to the Exclude List.

Exclude Application APIs

Classes, methods and functions that are part of an application API and typically called by a 3rd party cannot be obfuscated (as the obfuscated name cannot be predicted) and must be added to the Exclude list.

Header Tab

This tab is used to **append code and information** to the beginning of each encoded file. This allows you to *insert meta information* (e.g., copyright, version, etc.) into your encoded application. It also allows you to customize the message that is displayed if the Zend Optimizer is not installed.

The Header tab contains the following options:

Name	Description
Do Not Append Header Information	No Header information will be appended.
Append Default Zend Header Information	Appends the default Zend Guard Header information.
Generate Custom Header	<p>Append custom code and/or information to the header of each encoded file. The information (e.g., display copyright and version information) is embedded in the form of PHP comments and is viewable by the end-user. It also enables you to customize the message.</p> <p>The options are:</p> <ul style="list-style-type: none"> ▪ Generate Custom Header - This will include a customized header and override the default Zend Guard Header. ▪ Append Custom Header to Default Guard Header - This will include the custom and default headers. ▪ Text Header (select only one of these options): <ul style="list-style-type: none"> • Path - the location of a file containing a message or image to be displayed. • Text - write a header message. ▪ PHP Header (select only one of these options): <ul style="list-style-type: none"> • Path - the location of a file containing the code to be run. • Text - write code directly into the text box.



Adding PHP code can be useful when displaying custom error messages when the Zend Optimizer is not properly installed. For

example, to tell users to reinstall the product if Zend Optimizer is not properly installed, you can use **code** similar to the following:

```
print "ExampleApp is not properly installed. Please  
consult the User Manual, and reinstall it.";
```

Note:

The combined size of the comments and PHP code blocks is limited to approximately 62KB.

XML Editing Tab

This tab opens and displays the entire XML configuration file in editable format. All options can be viewed and edited manually, if required.

Note:

Please take human error into consideration when editing the XML file and check that there are no mistakes that could cause encoding to fail.



To generate an XML file:

1. In the General Tab click the Link "Export Parameters File" the "Parameters File" dialog will open.
2. Select a project from the list.
3. In the "Select the export destination" section name the file (without an extension correct: parameters incorrect parameters.xml) and click browse to specify the location for storing the file.
4. Click Finish to start generating the file.
5. Check the suggestions to be used, the others will be discarded automatically. **

Once the file has been created it can be run with the command line to secure PHP code. This can be done on the local machine or on another machine, see below for more information on how to encode PHP files with the XML file.

The XML editing option allows in one simple step, to encode projects that contain several encoding options. This includes different PHP versions and different levels of encoding. The XML is in essence a single file that contains different settings to be applied to different files and folders in the project.

When choosing the Option "Override Project Information" (located in the General Tab) the information collected in the "General Tab" is exported to an xml file. This file can then be used to encode code located on a different machine. The XML Editor included in the Zend Guard provides editing options to modify the XML file if necessary.

Encoding External code using the XML File

As we mentioned earlier, the XML output generated by the Zend Guard, can be used to encode files with the command line on this or on another machine.



To Encode using the command line (and the XML File):

1. Create the XML file.
2. If you are using a remote machine transfer the XML file and your **exclude list** file to the machine.
3. Move the XML file and the Encoder files located in the Zend Guard's installation directory (Zend\ZendGuard-<Your Product Version>\plugins\com.zend.guard.core.resources.<Your OS>\resources.
4. After arranging all the files activate the Encoder from the command line with the following command:

GuardEngine -- XML-FILE <Path to XML File>.

The settings defined in the file will be applied to the PHP code, including input and output directory locations. Therefore, you may want to manually edit the file if it is on a different machine to define the different file locations.

Encoding

Encoding with the Zend Guard allows developers to encode their PHP script at any time during the development process. More importantly, this enables code to be encoded prior to distribution or publishing.

Find out how to:

- **Use Encoding Features**
- **Encode projects** - encode files and expected outcomes.
- **Distribute Encoded Files** - transferring encoded files
- **Manually Encode** - encoding files without the user interface.

Encoding

Encoding with the Zend Guard allows developers to encode their PHP script at any time during the development process. More importantly, this enables code to be encoded prior to distribution or publishing.

Zend Guard optimizes secures and licenses PHP code to:

- **Expedite at Run-time**
Eliminates compilation and optimization at run-time.
- **Create Unreadable Source Code**
Files are encoded in an unreadable (to people) format.
- **Require Valid License** (User Configurable)
Files can be encoded to support or require licensing (License file restriction).
- **Create Expiration Date/Time** (User Configurable)
Files can be encoded to expire at a set date.
- **Create Encoded-Only Mode** (User Configurable)
Files can be set to cooperate with only associated encoded files that bear the same encoded signature.

Expedited at Run-time

Zend Guard optimizes PHP code, this results in faster execution and reduces the server's CPU load. The files are encoded and optimized, eliminating run-time compiling and reducing the number of run-time processing steps.

Secure

Zend Guard saves code in a closed Zend Intermediate Code format. This is a platform-independent binary code. It provides protection against tampering with the original source file, reverse engineering and copyright infringement. This is

the key to creating exclusive software solutions and protected commercial PHP applications.

License Requirements

These enable you to specify the license level which is then encoded into the file itself. There are three encoding options:

- **No Enforcement**

There is no interaction with a license file; no license is required in order to use the encoded file.

- **File Enforcement**

The file will not work unless a valid license file is available. The Licenser will generate the license.

- **License API Enforcement**

Uses the **zend_loader_file_licensed()** API function to verify valid license at specified points.

- **Product Name**

The product name, as referred to in the license files. When issuing a license for this application, you must use the same name.

Encoding determines the license level. Specific licensing details, such as the scope of a license, are determined during license generation.

No Obfuscation

Encoding is done by default even if there is no obfuscation method selected.

The option: "**Work only with Encoded Files**" requires that files be called only by other encoded files. This option offers additional protection from hacking and reverse-engineering.

The files are in a platform-independent format that can be deployed on all supported platforms with Zend Optimizer installed on their server. For details on supported platforms, see Zend's online system requirements at:

<http://www.zend.com/store/products/zend-guard-sysreq.php>.

Note:

Encoded files which have obfuscated local variables only are compatible with Zend Optimizer version 2.6 and above.

Encoded files which have obfuscated variables and also have obfuscated functions, classes and/or PHP Internal symbols require the latest version of Zend Optimizer (PHP4 Optimizer 3.0 and above; PHP5 Optimizer 3.3 and above).

The Zend Optimizer is available free of charge, from:


<http://www.zend.com/downloads>.

How to Encode

Once you have defined the Encoding Settings, you can encode and obfuscate files.



To Encode and Obfuscate files:

1. Click the Encode icon () , or
2. Right click the project in the Explorer and select **Encode Project**.

Results, warnings, and messages are reported in the Encoding Messages area during encoding. A brief explanation is included with each encoding error.

The Issuing Command message is a notification that a file was successfully encoded. The message includes the file name and the project settings details.

Note:

If you are using a beta version or an unregistered product, file will be encoded for 14 days only. In order to encode files for more than 14 days, the product **must** be registered.

Messages Area

If an error is found in the PHP during encoding, encoding execution will be terminated or ignored, as specified. A prompt to either correct the error or remove the file containing the error from the project will appear. Only projects containing error-free PHP files can be encoded.

If you have Zend IDE Client installed, double-click on the error message. The file that caused the message will open in the Zend IDE Client to the line of the script that caused the error. (For information on Zend IDE Client, refer to the Zend Website: <http://www.zend.com/products.php>.)

Stop encoding by clicking the **Cancel** button in the Encoding dialog.

Output

Output includes the *encoded* files, *any sub folders* containing files; and *copies all the files* located in the project folders.

It is important to consider the output of the path structures, such as sub-folders, when defining a Project. Individual files added to the project will be added to the target folder.

When a project contains multiple (sub)directories output will contain only unique directory structures. Directory structures common to all project directories will not be included with the output.



Example:

The directory **FinalRelease** was the only directory added to a project.

The following are the full path of the files below the *FinalRelease* directory:

1. C:\FinalRelease\Module1\Dialogs\NameSelector.php
2. C:\FinalRelease\Module1\Screens\Welcome.inc

For target path = C:\Products\ABCSoftware, the resulting output would be:

1. C:\Products\ABCSoftware\FinalRelease\Dialogs\Screens\NameSelector.php
2. C:\Products\ABCSoftware\FinalRelease\Module1\Screens\Welcome.inc

Distributing Encoded Files

Once all the files have been encoded, an organization can deploy the files in any way they choose (zip, tar and installer). As long as the encoder is installed on the same machine the product will work.

License restrictions, when applied will determine the period of time the application will be licensed for use.

The license files Zend Guard generates can be distributed separately.

Editing Encoded Files

Once the files have been encoded they are no longer editable. Any attempt to modify the files once encoded will result in rendering the files unusable. This provides an additional security layer for protecting intellectual property.

Note:

Files encoded by the Zend Guard are actually binary files. The proper protocol for transferring files between computers is as **binary files**.

All other transfer methods will corrupt the files and generate an error message.

Manual Encoding

If you prefer to configure and encode projects manually, without using the Zend Guard User Interface, use the file: **GuardEngine.exe**

Usage

You must prepare an **XML** file as the **input parameter** to the GuardEngine. This file must contain all required input information in order to properly encode and obfuscate your project.

Usage is as follows:

```
GuardEngine --xml-file <xml-file-path> [--export-candidates <target-file> <path1> [path2] ...]
```

Schema

The entire schema containing all parameters and format is included in the file: **Guard.xsd**. This file is located in the Zend Guard installation folder.

Obfuscation

Source Code contains various tags and names defined by the programmer. These names are typically made meaningful to make the code easy to understand and maintain, by developers.

Obfuscation *converts these tags and names into cryptic names*, in order to make the code difficult to understand by others, without affecting code execution.

Example:



Original Code:

```
<?php
function getName()
{
    echo "John Doe";
}

getName();
?>
```

After Obfuscation:

```
<?php
function a1234cd()
{
    echo "John Doe";
}

a1234cd();
?>
```

The function *getName*, when obfuscated, will be changed to something that does not have a meaning, such as **a1234cd** creating the following code:

As you can see from the example, the execution logic of the code is maintained, but the code has become difficult to understand.

Several options have been provided to suit various code protection levels. The Zend Guard obfuscation options support various PHP versions (including PHP 4 and PHP 5).

The following encoding and obfuscation options are provided through the Security tab:

- **Encoding** (no obfuscation)
- **Variables** - Converts user generated Variable names into machine-generated, cryptic Variable names. This completely scrambles the original context of the original, user generated Variable names.

- **Functions** - Converts user generated Function names into machine-generated, cryptic Function names. This completely scrambles the original context of the original, user generated Function names.
- **Classes** - Converts user generated Class names and methods into machine-generated, cryptic Class names. This completely scrambles the context of the original, user generated Class names and methods.
- **PHP Built in symbols** - Converts PHP language pre-defined names into machine-generated, cryptic names. For example *acos()*, *count_chars()*, *Exception*, *StdClass* and *echo* will be completely scrambled.

Important Note:

Obfuscation may change your original code to the extent that it may not execute properly.

Use the **Exclude List** to resolve such problems.

For example, code that calls functions referenced by string may not run after obfuscation:

```
<?php
function do_mysql_query($query) { ... }
function do_sqlite_query($query) { ... }

function executeQuery($dbname)
{
$query_function = "do_" . $dbname . "_query";
$result = $query_function("SELECT * FROM TABLE");
}
?>
```

After obfuscation the functions *do_mysql_query*, *do_sqlite_query* and *executeQuery* will be obfuscated and the value of *\$query_function* will **no longer** match any of the function names and a runtime error will occur (i.e. function not found error).

Therefore use the **Exclude List** to exclude the function names *do_mysql_query* and *do_sqlite_query* from being obfuscated.

Additional examples of functions, class names, methods and variables that should not be obfuscated can be found in "**Excluding PHP Entities**".

There is a direct correlation between the number of files obfuscated and the difficulty understanding and reverse engineering code. Therefore, complete project obfuscation will best protect your application.

Find out more about Obfuscation:

- [Encoding Only](#)
- [Variable Obfuscation](#)
- [Function Obfuscation](#)
- [Testing and Debugging](#)

Encoding Only

When no obfuscation options are selected, Zend Guard only converts PHP files into encoded binary files (Encoding) and does not require any developer involvement. Converting PHP files into encoded binary files makes PHP code unreadable by other developers.

Use this option when relatively low protection for source code is required.

Variable Obfuscation

A variable obfuscation modifies only **local variables** in source code. This provides improved security, is seamless to the developer and does not generate additional overhead. In addition, all obfuscated files are also encoded automatically.

Note:

The combination of Encoding, and obfuscating local variables, ensures that even if a third party does manage to decode the encoded files, they are prevented from exploiting the code.

Function Obfuscation

Function obfuscation, obfuscates more the application's **function names and calls**, (excluding entities that have been added to the Exclude List).

All obfuscated files are also **encoded** automatically.

Function obfuscation along with Variable obfuscation provides the most efficient security coverage for PHP code.

You can exclude specific entities from being obfuscated by means of the Exclude List.

Testing and Debugging Applications

Any code that has been changed or manipulated must be verified and checked to determine that it still works. Code that has undergone Zend Guard encoding/obfuscation is no exception.

No matter what type of encoding or obfuscation is applied to the code, it is necessary to validate the code by running a complete QA (Quality Assurance) cycle on the code. Code validation should be repeated after each time the code is encoded or obfuscated.

Extra attention should be given when obfuscating many types of entities. Errors found in the code indicate entities that should be included in the Exclude List.

Licenses

This section describes how to enforce license restrictions and create product licenses.

Find out how to:

- [Create a License](#)
- [Install a License](#)
- [Enforce a License](#)
- [Place a license](#)
- [Locate your Host ID](#)


Creating a License

This procedure describes how to create a license for your code along with descriptions of the different options and parameters available for a license.

Licenses are used in organizations in order to grant use of software to users who have legally obtained a License. In addition to usage, licenses can hold additional details pertaining to the number of users, license expiration and additional means for enforcing usage to authorized users only.



To create a license

1. Click the New icon () in the Zend Guard Toolbar and select **Zend Guard License** from the drop down menu. The Product License dialog will open. This is where the license is generated. (See [License Parameters](#) for descriptions of the

different parameters

Product License

Create Product License

[Gather](#) information from existing license file

Product Information

Product Name:

Register To:

Target File

Save License As:

Guard Private Key

Generated License Key 1 [Configure License Keys](#)

☐ Use License Key saved in external file:

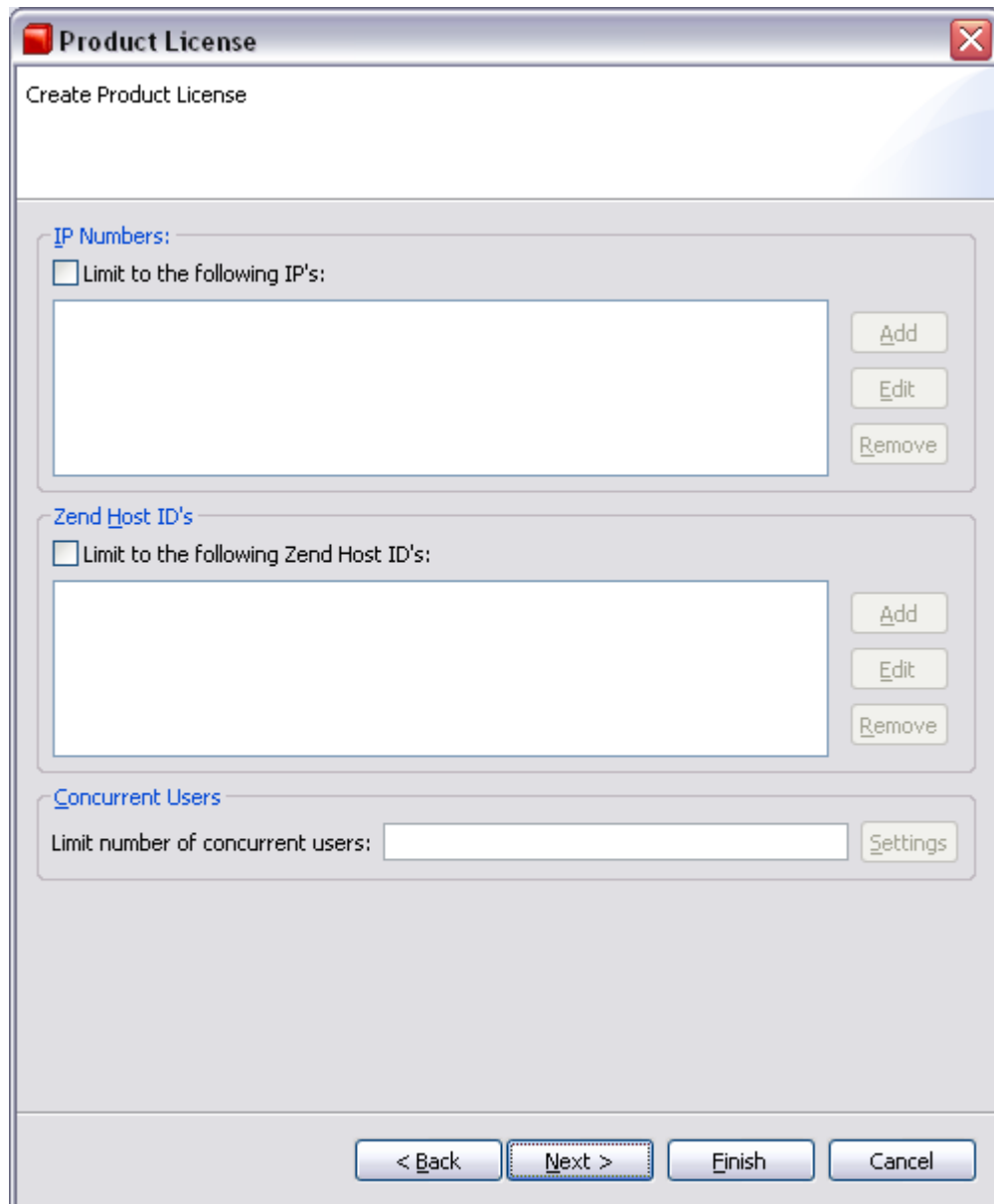
Product Expiration

☒ License never expires

☐ License expires on:

Figure 12 - Product License Dialog

2. Enter the relevant information click **Next**.
3. Use the following dialog to limit the allowed IP numbers, Zend Host IDs and the number of concurrent users.



The image shows a 'Product License' dialog box with a title bar containing a red close button. The main area is titled 'Create Product License'. It contains three sections: 'IP Numbers', 'Zend Host ID's', and 'Concurrent Users'. Each section has a checkbox to 'Limit to the following' followed by a list box and 'Add', 'Edit', and 'Remove' buttons. The 'Concurrent Users' section has a text input for 'Limit number of concurrent users' and a 'Settings' button. At the bottom are '< Back', 'Next >', 'Finish', and 'Cancel' buttons.

Product License

Create Product License

IP Numbers:

☐ Limit to the following IP's:

Add Edit Remove

Zend Host ID's

☐ Limit to the following Zend Host ID's:

Add Edit Remove

Concurrent Users

Limit number of concurrent users:

Settings

< Back Next > Finish Cancel

3. In the Concurrent Users section, click **Settings** to configure the number of **Concurrent Users**. Only after entering a value in the "Concurrent Users" section will the Concurrent Users Settings dialog be enabled. Click **Next** to continue

Note:

Concurrent user licenses will not work on PHP running in CGI mode.

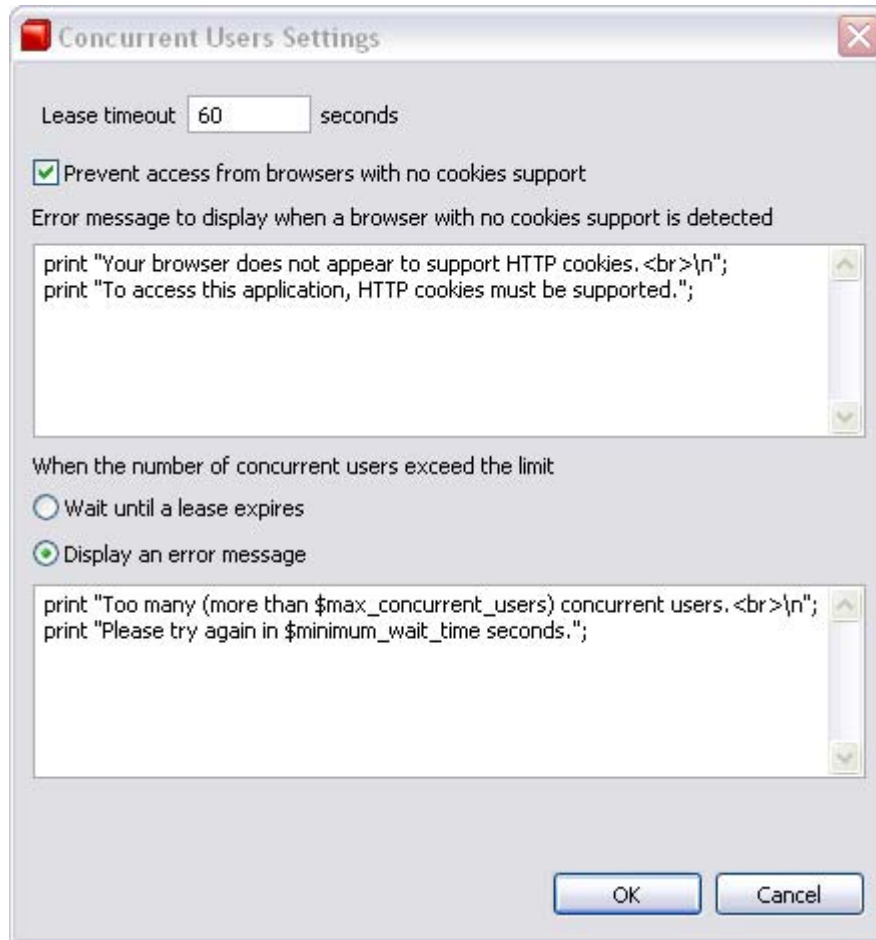


Figure 13 - Settings Dialog

4. Select and configure the following options, as required, then click

Next:

- Lease Timeout - The maximum amount of time that the end-user can leave the application idle before the license considers him/her to have ceased use of the application and disconnects the user from the application.
- Cookie Support - Prevents Access when the browser does not support cookies.
- Error Message - You can edit the Error Message sent to users without cookie support.
- Limit Exceeded - The actions to take when the number of users attempting to use the application exceeds the maximum allowed number.
- Wait until a lease expires - Keeps the user in queue on a first-come-first-serve basis until a lease becomes available.
- Error Message - You can create/edit a message to display

when the limit is exceeded.

5. The **Additional Information** dialog will appear. It is used to store additional information in the license file that can be used in conjunction with the function: `zend_loader_file_licensed()` in order to implement *customized licensing* rules. Information placed will be signed. If this information is changed, the license becomes invalid.

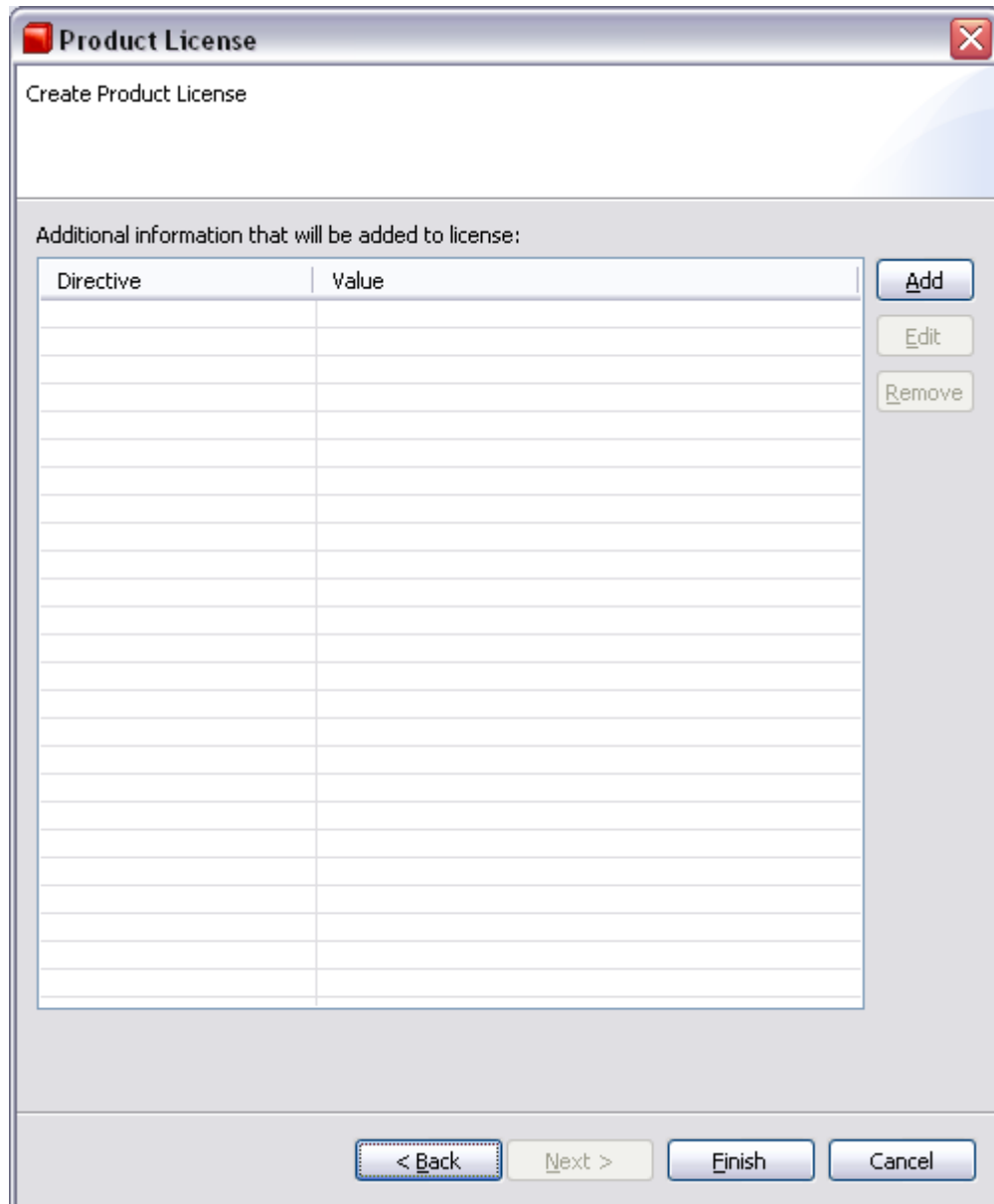
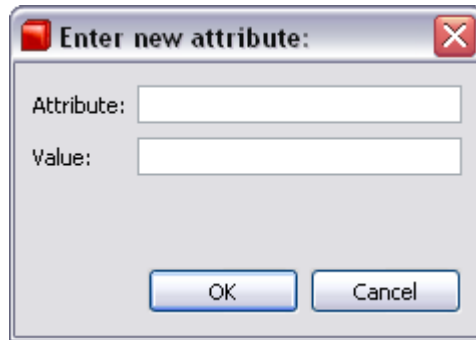


Figure 14 - Additional Information/Attributes

- Click **Add** to open the New Attribute dialog. Both the Attribute and its value must be filed in. Enter the information and click **OK** to return to the Additional Information dialog.
Repeat for each attribute.



New Attribute

7. Click **Finish** to create the license file.

A confirmation request will notify if the license generation succeeded or failed.

The new license file will be generated to the location specified in the Target File location (the first screen in the Create License Wizard). The license is now ready to be installed

Cookie Support

When the **Require Cookies** option is NOT selected, any page can be loaded, concurrently, **once**, from any number of browsers, even when the number of concurrent users is limited (even to a single user).

This happens because, **the first time that the page is accessed**, no cookie is present.

The second time that the page is accessed, a cookie ***is*** present (even though not required). The server then limits access.

In addition, if the "Require Cookies" option is **NOT** selected, **AND** the browser is set to **not** receive cookies, any page can be loaded, *any number* of times, from *any number* of browsers.

This is because no cookie is present at **any** time. The server has no way to know that the page has been accessed.

In order to effectively limit the number of concurrent users you **must** require cookies.

Limit Number of Concurrent Users

This creates a "floating license" that limits the number of users who can concurrently use the application. When the number of users attempting to use the application exceeds the maximum, you can create the Error Message that "excess" user receives. You can also select the **"Wait Until a Lease Expires"** option. This option keeps the user in the queue on a first-come-first-serve basis until a lease becomes available.

Example: The developer sets the number of permitted concurrent users as 20. As long as there are no more than 20 requests to use the application, access to the application is granted to all users. If, however, there is a 21st user, he/she is not allowed access to the application.

Lease Timeout

Lease Timeout eliminates inactive users and frees leases for queued users.

Caution:

Changing license file information (that does **not** begin with X-) will invalidate the license.

License Parameter	Explanation
Product Name	<p>The name of the product. It is used to match licenses with encoded files and therefore must match the Product Name given to the encoded files.</p> <p>When a license-check is performed, all licenses found in the license directory are reviewed by a license-signature that includes the product name.</p> <p>Only a license that matches by product can determine license rights and information.</p>
Registered To	<p>The name of the licensee i.e. the registered owner. This data (and other) is used to generate the license; it is included in the array returned by the Zend Guard API on valid licenses.</p>

License Parameter	Explanation
Additional Information	<p>Used to store additional information in the license file.</p> <p>Information placed here will be protected by Zend Guard's digital signature technology. <u>If this information is changed, the license becomes invalid.</u></p> <p>Information is stored in using the format 'directive = value' (e.g., Shareware=yes). These lines appear as text in the license file and are accessed by the same means as accessing a text file. In addition, if a license is valid, the Zend Guard API function <code>zend_loader_file_license</code> returns an array containing these values.</p> <p>This information is used in conjunction with the function: <code>zend_loader_file_licensed()</code> to implement customized licensing rules. For example, you can disable certain features if the "Shareware = yes" entry exists in the license file.</p> <div style="background-color: #e6f2ff; padding: 10px; margin-top: 10px;"> <p>Note:</p> <p>To store information that will NOT be protected by the digital signature mechanism, begin the name of the directive with the prefix: X-. For example, 'X-Create-On = December 28, 2006'. Lines beginning with X- will be ignored when the license signature is generated, however these lines will be returned by <code>zend_loader_file_license</code> when the license file is determined valid.</p> </div>
Lock to Zend Host IDs	<p>Locks a file to a specific Zend Host ID(s). This code uniquely identifies a specific machine. Multiple entries should be separated with spaces.</p>
Limit to the Following IPs	<p>Adds IP information, to bind the license to a specified IP or IP range.</p> <p>IPs can be specified as single IPs or a range by using wildcards, for example:</p> <p>10.1.1.17</p> <p>10.1.*.*</p> <p>10.1.255.255</p> <p>10.1.3.0/24</p> <p>Note: To specify multiple IPs use commas as separators.</p>

License Parameter	Explanation
Limit Number of Concurrent	<p>Limits the number of users who can simultaneously use the application.</p> <p>Developers can set the limit while writing the application. The license is encoded and forms part of the encoded application file given over to the client.</p> <p>Concurrent User Settings:</p> <p>Lease timeout - The maximum amount of time that the end-user can leave the application idle before the license considers him/her to have ceased use of the application and disconnects the user from the application.</p> <p>Prevent Access from Browsers With no Cookie Support - Deny access from browsers without cookie support and display a predefined error message. Browsers that do not have HTTP cookie support cannot be detected by the Concurrent Users mechanism.</p> <p>Generally, you should use this setting with intranet applications when access to the site deploying the application is limited and you can control the users' browser settings. This will prevent browsers without cookie support from access to the application. The setting will cause a slight performance penalty on initial access to the application. It is not generally noticeable in most intranet environments unless the internet connection is slow. You should not use this setting with internet applications when access to the site deploying the application is unlimited and you cannot control the users' browser settings.</p> <p>When The Number of Concurrent Users Exceeds Limit Options -Wait until a lease becomes available and/or display an error message.</p>

License Files

A license file contains elements that control whether your product may run on the end-users' machines, how many users may use the product simultaneously, and on which machine or ID address, etc. The license File contains digitally signed data, settings and user defined information.



Example 1:

```
Product-Name = Drink
Registered-To = Bob
IP-Range = 10.1.*.*
Hardware-Locked = Yes
Host-ID = H:MFM43-Q9CXC-B9EDX-GWYSU
Host-ID = M:3QNKS-WPDD5-B3WU6-EFVZH
Expires = 09-Oct-2005
Tea = Mint Flavor
X-Coffee = Black with no cream or sugar
```

Example 2:

```
Product-Name = My Product
Registered-To = ABC Company
Hardware-Locked = No
Host-ID = Not-Locked
Expires = Never
X-Home0 = Region 1 Priority
X-Home1 = Region 8 Dismissed
X-Home2 = Region 12 will update next run
X-NextAppointment = See X-Home0
X-LastAppointment = See X-Home1
X-PromptRemoder = Hire a Janitor
ViolatedMyLicenseRules = False
```

Example 3

```
Product-Name = HIJK_Intranet
Registered-To = HIJK Company
IP-Range = 10.1.*.*
Hardware-Locked = No
Host-ID = Not-Locked
Expires = 02-Jul-2002
X-UsersOnline = GinaK,RO,RowlingsC,Janice,Temp1,SysAdmin1
X-UnauthorizedCode = G5gH7^&*KkJ200 by User ICNeilW at 15:30
June1
```

```
IllegalAccessDetected = False  
HackerDetected = False  
VirusDetected = False
```

Installing a License File

The following procedure describes how to install a license file that was generated by Zend Guard on a user's machine. This procedure should be done as part of the installation/transition from evaluation process of your application.



To install a license file:

1. Open the **php.ini** file in a text editor.
2. Locate the line with the directive **zend_optimizer.license_path**.
If a line for this directive does not exist, add a line to the php.ini and type **zend_optimizer.license_path=**
3. If the path where the license file resides is not found on the **zend_optimizer.license_path** directive-line (after the = sign), add it to the end of the line. Remember to separate the paths with a colon (for UNIX) or a semicolon (for Windows).
4. If your PHP is installed as SAPI/ISAPI, restart the Web-server for the changes to take effect.

Note:

Placement of the license file is critical to the validation process. If the license is not in the correct location, Zend Guard will assume that the product is **not-licensed**. License validity and locations are checked and loaded when PHP starts up.

License Enforcement

License restrictions control the status of a license file. License files are checked for validity when the PHP server starts. All valid license information is then stored in the memory of the License Registry. Invalid licenses are also registered. This allows the application to check for the type of error that has occurred.

- For **License File Enforcement** files, the Zend Optimizer checks and then allows or prohibits its running of the encoded file.
- For **License API Enforcement** files, you must program the call for a license lookup using the Zend Guard API and implement your policy based on whether a valid license is available or not.

License File Location

The location of license file must be written into the **php.ini** file. This enables it to be validated and loaded into the license registry if valid. Depending on your setup you may need to restart the web server. For example: SAPI, ISAPI configurations that run a persistent PHP module require restarting the Web server to apply changes to the PHP engine.

Licenses are either valid or not-valid. The state "**No license found**" is treated as an invalid license. Therefore, proper license installation is necessary.

You can define specific files or license file directories to contain the license. If you specify a license directory, all files with the file extension **.zl**, are checked for validity. If valid, they are loaded into the license registry, once, at the startup of the PHP server.

Files encoded to check for valid licenses will check the license registry for a license matching its product specification.

After updating the php.ini restart the server to apply changes for SAPI/ISAPI configurations.

The directive in the php.ini file for license paths is

zend_optimizer.license_path. The syntax is as follows:

zend_optimizer.license_path=LicensePath1:LicensePath2

Where: *LicensePath* is the path to the file or directory holding the correct license file. For UNIX, multiple paths are entered separated by colons (colon delimited.) For Windows, multiple paths are entered separated by semicolon (semicolon delimited.)

Examples:



The following lines specify two license files (UNIX).

zend_optimizer.license_path=/usr/local/Zend/licenses/Lic.zl:/usr/local/Zend/licenses/Lic2

The following line specifies one license file and a license folder (Windows).

zend_optimizer.license_path=C:\dir1;C:\dir2;C:\dir3\lic.zl

Zend Host ID

The Zend Host ID is used to generate licenses locked to a specific machine (hardware).

In order to obtain the Zend Host ID you must copy and run the zendid.exe program on the machine for which the license is to be issued.

The zendid.exe program can be found with the files of the Zend Guard.

Getting the Zend Host ID

The following procedures describe how to get your Zend host ID in Windows and Unix (includes Linux and Mac).



To get your Zend Host ID in Windows:

1. Copy the zendid.exe application to the users computer.
2. Open the Windows Start menu, select Run.
The Run dialog box will open.
 - For Windows NT, 2000, or XP operating systems, type **cmd** and click **OK**.
 - For Windows 95, 98, or ME operating systems, type **command** and click **OK**.The DOS Command Shell will open.
3. In the shell application type **zendid.exe** and press **ENTER**.
The Zend Host ID will be printed to the screen (as shown in the example below).
4. Record the Zend Host ID code.
5. Click **Exit** and press **ENTER**.
The shell application will close.

Use this ID number to generate a license for a specific machine. To see how to generate licenses see "[Creating a License](#)".



Example:

In this example the Host ID is- H:MFM43-Q9CXC-B9EDX-GWYSU In the shell application type:

```
C:\WINDOWS\Desktop>zendid.exe
H:MFM43-Q9CXC-B9EDX-GWYSU
C:\WINDOWS\Desktop>
```

**To get your Zend Host ID in Unix:**

1. Copy the **zendid** application to the users computer.
2. Open a terminal and run the command **zendid**.
The Zend Host ID prints to the screen as shown in the example below.
3. Record the Zend Host ID code.
4. Click **X** in the upper corner of the active screen to close the shell application.

Use this ID number to generate a license for a specific machine. To see how to generate licenses see "[Creating a License](#)".

**Example:**

In this example the Host ID is- M:DRMTW-59QCX-B9EDX-12BM In the shell application:

```
bash# ./zendid
M:DRMTW-59QCX-B9EDX-12BM
bash#
```

License Architecture and Behavior

The following topic describes how licenses are identified, what the changes are in comparison to the previous version (Guard 4), license options, restrictions and usage.

Run Flow

Software that is distributed with license restrictions depends on the Optimizer component to enforce restrictions. When the Optimizer is loaded, it finds and loads all the licenses (.zl files) in the license directory as specified in the "zend_optimizer.license_path" directive (in your php.ini).

When a file is encoded with a license (using --license-product or sign-product) the Optimizer tries to find a matching valid license.

There are certain conditions for a license to be valid.

It needs to be:

- Produced with the same private key that was used to encode the file.
- Have the same product name that was specified during the encoding.

Both restrictions must be satisfied. (Restrictions are checked on loading, so if they have expired while the web-server was running, they will still be accepted.

Key Management

Up until Guard 5 (exclusive) private keys were produced by Zend, and were written in the Guard License. When using an evaluation version (no license) the application used a hard coded private key. However, this meant that in order to produce a license for an application, the users had to use the same Guard license that they used to encode the script. Moreover, when users upgraded the Guard version, they had to contact Zend in order to get a license with the same private key they had.

In Guard 5.0 private key was separated from the license by changing the license generator (zendenc_sign) to support:

- Key generation
- Key extraction (from an old license)
- Support key verification

The result is that when encoding file, and producing a license, a key must be specified.

This change allows users to manage keys on their own, without the need to contact Zend. Therefore, the Guard license is no longer part of the licensing process.

For example: Users can now encode their files on one machine (using one Guard license) and produce a license on other machine (using a different Guard license) as long as they use the same key.

License Options

Guard offers two levels of licensing support:

1. License is required (using --license-product switch)
2. License is supported (using --sign-product switch)

The difference is the script that requires a license will not run unless a valid license is present. Script that supports a license will run without a valid license, but, if a license is present, it can be accessed (using the **Optimizer API functions**). This enables you to define your own restrictions.

For example: An application can be made to run in evaluation mode without a license, in limited mode, with one type of license and in full mode with another.

License Restrictions

Guard licenses can be enforced based on the following categories:

- Date
- Concurrent Users
- IP Range
- ZendID (unique code based on the machine hardware)

In addition, users can add custom fields (in key=value format), and write restriction code.

Usage

The following procedure describes how to encode a project using a private key:
If you do not have a valid private key file, generate/extract one first before encoding your project.



To encode a project using a private key:

1. Use `--genenckey <path_to_key_file>` to generate a Key or `--extractkey <license_file> <path_to_key_file>` to extract a key.
2. Encode the application with the desired Licensing support.
3. Specify the path to your private key. (for example, `zendenc --license-product foo --private-key myKey foo.php foo.enc.php`).
4. Generate the license (`zendenc_sign myLic myKey`. This will run a textual wizard)
5. Copy the license to the license path (defined in the `php.ini` file)
6. Restart the web server to load the license

The files will be encoded with the restrictions set in the license.

Tips and Tricks

- To get a field from the license (for example, for custom limitation) use the `zend_loader_file_licensed` API function
- To load license without the need to restart the web server use the `zend_loader_install_license` API function
- License generator (`zendenc_sign`) usage can be found [Zendenc_sign_usage here] (Or by running it with no parameters)
- Other API functions can be found [Optimizer_PHP_API_functions here]
- Other ini directives can be found [Zend_Optimizer%27s_Php.ini_directives here]

Command Line

The processes described in this section describe how to manually set configurations for encoding and license generation.

Note:

The option to **generate an XML configuration file** was introduced to replace the need to use the command line for encoding projects. Using the XML file option instead of the command line creates a portable file containing your encoding and license settings. This file can be easily distributed to other servers to provide a unified way to quickly apply your preferences to multiple environments.

Depending on user preferences, encoding and license generation can be done using the command-line to run the following commands inside the command shell.

- **zendenc**

The command-line version of Zend Guard for encoding PHP 4 files. (PHP 5 can be used as long as they are PHP 4 compatible i.e. do not use the new features introduced in PHP5).

- **zendenc5**

The command-line version of Zend Guard for encoding PHP 5 files.

- **zendenc_sign**

The command-line version of Zend Guard's licensing tab for creating a signature license file from a license definition file.

These binaries are located in the plugin called

"com.zend.guard.core.resources.<OS version>_5.0.x and can be referenced from there.

The plugin is located in your installation directory under: <install_dir>\Zend\Zend Guard -5.0.x\plugins\com.zend.guard.core.resources.<OS version>\resources.

zendenc and zendenc5 – Command Line

This section serves as a technical reference to the zendenc and zendenc5 commands.

Command Description

Command **zendenc + zendenc5**

Synopsis Command-line function for encoding files.

Syntax 1 zendenc [options] SourceInputPath [EncodedOutputPath]

Arguments **SourceInputPath**

The path and/or file name of the source directory or file. This must be a valid full or relative path. This is a mandatory argument for all encoding operations. The command line syntax requires the SourceInputPath parameter precede the EncodedOutputPath parameter.

EncodedOutputPath

The path and/or file name of the target file name where the encoded file is written. This must be a valid full or relative path.

This argument is not required for encoding operations using the --delete-source and --rename-source options.

The command line syntax requires the EncodedOutputPath parameter (when used) is entered following the SourceInputPath parameter.

--option^x [option_parameter^x]

Various options can be entered to control the functionality of zendenc.

Options may have parameters that immediately follow the option.

Every option must be preceded by a double dash -- prefix

Command Option - Syntax

Syntax	Description
--optimizations	Optimization mask. (default value: [+++++++])
opt_mask	opt_mask is an integer representing a bit-mask. The default value enables all of the optimization passes. Each optimization pass of the Zend Optimizer can be turned on or off based on the mask entered.
--encoded-only	Force cooperation with other encoded files only. This option generates files that work exclusively with associated encoded files. Associated encoded files are those generated by the same company. Files that do not share the same encoded company association cannot call these files.

--asp-tags on/off	<p>Turn ASP tag ("<%") recognition on/off. (default: off). On or off must be specified as an argument when using this option.</p> <p>The default, when this option is not used in the command-line, is - off.</p>
--short-tags on/off	<p>Turns short PHP tag ("<?") recognition either on or off. On or off must be specified as an argument when using this option.</p> <p>The default, when option is not used in the command-line, is - on.</p>
--no-header	<p>Disables the PHP-compatible header that is added to the top of every encoded file by default. Encoded files generated with this option will not display a meaningful error when loaded by PHP that doesn't have the Zend Optimizer properly installed. Using this option saves approximately 1.5KB for every encoded file. Do not use it unless disk space constraints are critical.</p>
--prolog- filename <file>	<p>Embed the information in the specified file into the header of the encoded file (overrides --no-header)</p>
--delete-source	<p>Permanently deletes (see warning below) the original source files specified in the SourceInputPath and saves the encoded files in its place.</p> <p>This option has no option parameter.</p> <p>When this option is use, do not use the output_file parameter.</p> <p>Warning:</p> <p>To avoid permanent loss of non-encoded scripts, make a backup. Deleted files cannot be restored or recovered and will be permanently deleted with this option. If you are unsure about deleting the source files, use the --rename-source option instead.</p>
--rename-source <ext>	<p>Move the original source file to <input_file>.<ext> and save the encoded file in its place. The output_file parameter should not be specified when using this option.</p>
--recursive	<p>Encode files in directories, recursively. input_file and output_file are the source and target directory names.</p>
--php-only	<p>In recursive mode, don't copy non-PHP files from the</p>

	source to the target.
--ignore-file-modes	Do not preserve ownership, permissions and timestamps for encoded files (preserved by default).
--include-ext <ext>	<p>Encode files with this extension in recursive mode. By default, the following extensions are encoded: inc php ihtml php3 php4.</p> <p>This option can be entered more than once to include multiple files.</p> <p>ext is the file extension of the files, which will be included in the encoding source files.</p>
--exclude-ext <ext>	<p>Used in conjunction with the --recursive option to remove file extensions from the default extensions This option can be entered more than once to exclude multiple files.</p> <p>ext is the file extension of the files, which will be excluded from the encoding source files.</p> <p>Don't encode files with this extension in recursive mode.</p>
--exclude-file <name>	Don't encode this file in recursive mode. File should be FULL PATH !!!
--no-default-extensions	Don't automatically use the predefined extensions. Only extensions added with --include-ext will be encoded.
--ignore <pattern>	<p>Files matching this pattern will be ignored in recursive mode. Default pattern list: [empty]</p> <p>This option can be entered more than once for ignoring multiple files and can be used to ignore a subfolder by specifying the full path in recursive mode.</p>
--ignore-errors	<p>Continue encoding additional files even if encoding one of the files fails in recursive mode.</p> <p>Used when encoding multiple files. If an encoding error occurs while encoding a file, the Zend Guard continues processing the other files.</p>
--quiet	<p>Reports names and errors only. Does not report the progress of the encoding or messages other than errors.</p> <p>This option does not have an option parameter.</p>
--silent	<p>Reports errors only. Does not report names, progress, or messages other than errors.</p> <p>This option does not have an option parameter.</p>
--force-encode	Allow encoding previously encoded files. (NOT

	recommended!)
--expires <yyyy-mm-dd>	Make an encoded file to expire on the given date. Date is in yyyy-mm-dd format.
--license-product <name>	<p>Encodes files to only work with a valid license for the ProductName specified (encoded into a signature).</p> <p>ProductName must exactly match the Product Name entered when generating a license.</p> <p>This is the same as the License file restriction setting.</p>
--sign-product <name>	<p>Encode files with the product name signature. Same as the Support Licensing feature, which works with the Zend Guard API function to identify if a valid license exists. (Scripts check for this signature.)</p> <p>ProductName must exactly match the Product Name entered when generating a license.</p>
--use-crypto	Cryptographically sign scripts to prevent unauthorized modification.
--obfuscation-level <lvl>	Set the default obfuscation level. The default is 0 - no obfuscation
--export-list <file>	The file that holds the function obfuscation export list. (The file format is one function name per line)
--export-candidates <file>	Create list of functions recommended for exporting in this file.
--export-php	Automatically export all internal (PHP) functions.
--help	<p>Displays help about encoding options.</p> <p>This option is entered as follows:</p> <p>zendenc --help</p>
--symlinks	Copy symlinks as symlinks, do not try to resolve them.
--private-key <file>	encodes files with the private key in <file> The file should look like: Company-Key = <private key>
--obfuscate-stats	Result is like 1/16 3/15 1/1 3/15. The groups are: variables functions classes methods. Each group has form of X/Y, where Y is the total count of these entities, and X is how many of them were not obfuscated. For variables, it means how many of the functions were not obfuscate, i.e. 1/16 means that 1 of 16 functions were not obfuscated with level 1. For others, it means how many names weren't
Obfuscation statistics	

obfuscated - i.e. 3/15 means 3 of 15 functions, 1/1 means 1 of 1 classes and 3/15 means 3 of 15 methods (Note - functions include methods).

zendenc - Command Line Examples

The following are examples, which use the command-line zendenc and its options. Each begins with the command line and is followed by an explanation of its results.



Example 1

```
/usr/local/Zend/zendenc --quiet /WorkFolder/mysource.php  
/EncodedFolder/myencoded.php
```

Result: The file /WorkFolder/mysource.php is used as the source of the encoded PHP file myencoded.php. During encoding, only errors are reported. The output is an encoded file written to /Encoded Folder/myencoded.php. The source file remains unchanged.

Example 2

```
/usr/local/Zend/zendenc --rename-source old  
/WorkFolder/ReplaceMe.php
```

Result: The /WorkFolder/ReplaceMe.php file is renamed by adding “.old” as a file extension. The new name of the source file is ReplaceMe.php.old.

Output is an encoded file written to the same name as the original source file.

The result is file /WorkFolder/ReplaceMe.php is an encoded file and the file /WorkFolder/ReplaceMe.php.old is the un-encoded source.

Example 3

```
/usr/local/Zend/zendenc --delete-source /WorkFolder/mysource.php
```

Result: The file /WorkFolder/mysource.php is used as the source then deleted and is replaced by an encoded file of the same name.

The result is that the file /WorkFolder/mysource.php is an encoded file and the original un-encoded content has been deleted.

Example 4

```
/usr/local/Zend/zendenc --asp-tags on mysource.php myencoded.php
```

Result: The file /WorkFolder/mysource.php is used as the PHP and

ASP source file to be encoded. ASP tags are encoded along with the PHP thanks to the fact that ASP tags are treated as PHP code. The resulting output is an encoded file written to /Encoded Folder/myencoded.php. The source file remains unchanged.

Example 5

```
/usr/local/Zend/zendenc --short-tags on --expires 2005-01-01 --recursive --ignore-errors --license-product SuperWebGame --include-ext htm --exclude-ext phtml --ignore "a*" /WorkFolder1/ /WorkFolder2/
```

Result: Encodes files in /WorkFolder1/ and below with the file extensions: inc php htm php3 php4 but not phtml or files beginning with the letter "a". Encoding will be processed for short tags and each encoded file will both expire on Jan. 01 2005 and will require a valid license for the product name SuperWebGame to work. The output files and any sub structures will be written to /WorkFolder2/. Lastly, the encoding will continue to process all files and will not stop the encoding process because of encoding errors.

Creating a Signature License (Command Line)

Signature license files work in conjunction with the following encoded files:

Require Valid License or **Licensing Support**. In addition, they must be encoded from the same product.

This can be an encoded file created by the Zend Guard user interface, or an encoded file generated by **zendenc** with the **--license-product** or **--sign-product options**.

There are two ways to create a signed license file:

- Create a License Definition File, and run `zendenc_sign <lic_def_file> <signed_file>`
- Run `zendenc_sign <signed_file>`. The signing utility will then prompt you to type in the license information.

License Definition File

The following table lists the license definition parameters, a short description, rank, and required status, as well as examples. All of the following are case sensitive. Every line has to be written in the format:

Entry = Value

(The format is a single space before the equal sign and another one after the equal sign). The order of definitions will be a permanent part of the signature. This means the lines of the license file must maintain the same sequence. If the order of the definitions in the license file changes from what the signature contains, the license becomes invalid. Definition Files can be created in text editors, such as Microsoft Notepad or VI.

Field Name	Description
Product-Name	<p>The name assigned to Product. This must be the same name used when encoding the PHP files.</p> <p>REQUIRED</p> <p>Example:</p> <p>Product-Name = Drink</p>
Registered-To	<p>The Name of the Registered owner of the license.</p> <p>REQUIRED</p> <p>Example:</p> <p>Registered-To = Bob</p>
Expires	<p>Expiration date of the license. Used if the license is issued with a date restriction.</p> <p>Format:</p> <p>DD-MM-YYYY</p> <p>OPTIONAL</p> <p>Example:</p> <p>Expires = 09-Oct-2005</p>
IP-Range	<p>Limits the use of the license to IP addresses that fall within specification. Supports wildcards for any of the IP place holders, as well as the two types of net masks (filters). Netmask pair An IP a.b.c.d, and a netmask w.x.y.z. (That is., 10.1.0.0/255.255.0.0), where the binary of mask is applied to filter IP addresses.</p> <p>ip/nnn (similar to a CIDR specification) This mask consists of nnn high-order 1 bits. (That is, 10.1.0.0/16 is the same</p>

as 10.1.0.0/255.255.0.0). Instead of spelling out the bits of the subnet mask, this mask notation is simply listed as the number of 1s bits that start the mask. Rather than writing the address and subnet mask as 192.60.128.0/255.255.252.0 the network address would be written simply as: 192.60.128.0/22 which indicates starting address of the network and number of 1s bits (22) in the network portion of the address. The mask in binary is (11111111.11111111.11111100.00000000).

OPTIONAL

Example (Wildcard):

IP-Range = 10.1.*.*

Example (Net Mask):

IP-Range = 10.1.0.0/255.255.0.0

Example (Net Mask):

IP-Range = 10.1.0.0/16

Host-ID

Coded string (Zend Host ID) used to lock the license to a specific hardware. The Zend Host ID obtained from the machine where the encoded files and license are to be installed. The Zend Host ID code can be obtained by using the zendid utility. For more details, see Getting the Zend Host ID on page 47.

REQUIRED if Hardware-Locked is set equal to YES.

Meaningless if Hardware-Locked is set equal to NO.

Example:

Host-ID = H:MFM43-Q9CXC-B9EDX-GWYSU

Hardware-Locked

[YES | NO] option that indicates if the license will be locked to a specific machine using the Zend Host ID code(s). If set to YES, the Host-ID is required.

REQUIRED

Example:

Hardware-Locked = YES

UserDefinedField

OPTIONAL

Example:

Tea = Mint Flavor

X-

UserDefinedField

User defined field prefixed with X-. Any additional information, which the user would like to add to the license

file but not to the signature. This information allows flexibility of input both prior to generating the signature and after. Any change to this information in the signature license-file will have no impact on the validity of the license.

OPTIONAL

Example:

X-Coffee = Black with no cream or sugar

zendenc_sign – Command

This section serves as a technical reference to the zendenc_sign command.

Command **zendenc_sign**

Synopsis Command-line function for creating a signature license file from a license definition file.

Syntax 1 zendenc_sign LicenseDefinitionFile LicenseFileName

Arguments LicenseDefinitionFile - The file containing the source names and information in an equation format, FieldName = FieldValue (for more information on how to create a license definition file, see License Definition File in the preceding section)
LicenseFileName - The name and extension to be given to the generated signature license file.

The zendenc_sign Commands are:

--version - get version number.

--genenckey <path_to_key_file> - generate user key for the Encoder put the private key in <path_to_key_file>.

--extractkey <license_file> <path_to_key_file> - take the private key from <license_file> put the private key in <path_to_key_file>.

--verify <license_file> <path_to_key_file> - check if the license in <license_file> is good for the private key.

[source_file] result_file <path_to_key_file> - signs <result_file> license file with optional paramter file <source_file>, with the private key in <path_to_key_file>.

The following is an example of a License file:

**Example Signature License File:**

```
Registered-To = ABC Company
Hardware-Locked = No
Host-ID = Not-Locked
Expires = Never
X-Home0 = Region 1 Priority
X-Home1 = Region 8 Dismissed
X-Home2 = Region 12 will update next run
X-NextAppointment = See X-Home0
X-LastAppointment = See X-Home1
X-PromptRemoder = Hire a Janitor
ViolatedMyLicenseRules = False
Verification-Code =
wCcJ6mJMYH7vggYkS3m+3/dUL332aHRy0xtYnc55CC
TCtcxWXvuU5lOG4w==
```

Zend Guard API

With the Zend Guard API, you can complete the following tasks:

- Check if Zend Optimizer is enabled to handle encoded files.
- Check if a valid license exists and gather information from a valid license.
- Get the running files path at runtime.

The following API describes how the Zend Guard determines if encoding is enabled:

Name	zend_loader_enabled
Synopsis	Checks the Zend Optimizer's configuration to verify that it is configured to load encoded files.
	<p>Note:</p> <p>Zend Optimizer setting can be configured in the php.ini file. Enable Optimizer line syntax: zend_optimizer.enable_loader = on off.</p> <p>By default, this is set to On. Therefore, you do not have to make any changes for the Optimizer to run encoded scripts.</p>
Syntax	zend_loader_enabled()
Results	<p>Returns Boolean</p> <p>TRUE The Optimizer is configured to load encoded files.</p> <p>FALSE The Optimizer is not configured to load encoded files.</p>

Name	zend_get_id
Synopsis	Returns array of the host ids. If all_ids is true, then all IDs are returned, otherwise only IDs considered "primary" are returned.
Syntax	array zend_get_id([bool all_ids = false])

The following API describes how the Zend Guard checks if a valid license exists:

Name	zend_loader_file_licensed
Synopsis	<p>Compares the signature of the running file against the signatures of the License files loaded by the php.ini file into the License Registry. If a valid license file exists, the values of the license file are read into an array.</p> <p>If a valid license does not exist or is not specified in the php.ini, it will not be entered in the PHP server's license registry.</p> <p>If a valid license, matching in product and signature cannot be found in the license directory, an array is not created.</p> <p>For information on the proper installation of a license file as well as the php.ini directive, see Limiting the Number of Concurrent Users: Proper Use of Cookies</p> <p>When the "Require Cookies" option is NOT selected, any page can be loaded, once, from any number of browsers concurrently even when the number of concurrent users is limited (even to a single user).</p> <p><i>This happens because, the first time that the page is accessed no cookie is present. The second time that the page is accessed, a cookie <i>*is*</i> present (even though not required). The server then limits access.</i></p> <p>In addition, if the "Require Cookies" option is NOT selected, AND the browser is set to not receive cookies, any page can be loaded, <i>any number</i> of times, from <i>any number</i> of browsers.</p> <p><i>This happens because no cookie is present at any time. The server has no way to know that the page has been accessed.</i></p> <p>In order to effectively limit the number of concurrent users, you must require cookies.</p>
Syntax	\$lic_info = zend_loader_file_licensed();
Results	<p>Returns an array or FALSE array.</p> <p>If an array is returned, a valid license exists in the location indicated in the php.ini file and for the product. The array has an element for each line of the license file. This includes the license generation settings and any additional user information added to the license.</p> <p>Example from a license file:</p>

Name	zend_loader_file_licensed
	<p>Product-Name = My Product</p> <p>The array index names are the <i>line content</i> from the left side of the equations (Product-Name) within the license file text.</p> <p>The array values are the <i>text content</i> to the right of the equation (My Product.</p> <p>FALSE</p> <p>"False" means that no valid license was found. This can mean that no license exists in the license directory or that the license file exists and has become invalid or corrupted.</p>

Name	zend_loader_install_license
Synopsis	Dynamically loads a license for applications encoded with Zend Guard. The Override controls if it will override old licenses for the same product.
Syntax	boolean zend_loader_install_license(string license_file[, bool override])

The following APIs describe how Zend Guard checks for obfuscation:

Name	zend_loader_current_file
Synopsis	Obtains the full path of the currently running file at run-time, in other words, the path of the file calling this API function. Does not evaluate the running files path during encoding, but evaluates only at run-time.
Syntax	zend_loader_current_file()
Results	Returns a string containing the full path of the currently running file.

Name	Zend_obfuscate_function_name
Synopsis	Obfuscate and return the given function name with the internal obfuscation function.
Syntax	string obfuscate_function_name(string function_name)
Results	Returns the obfuscated form of the given string.

Name	Zend_obfuscate_class_name
Synopsis	Obfuscate and return the given class name with the internal obfuscation function.
Syntax	string obfuscate_class_name(string class_name)
Results	Returns the obfuscated form of the given string.

Name	zend_loader_file_encoded
Syntax	boolean zend_loader_file_encoded()
Results	Returns true if the current file is a Zend-encoded file.

Name	zend_current_obfuscation_level
Syntax	int zend_current_obfuscation_level()
Results	returns the current obfuscation level support (set by zend_optimizer.obfuscation_level_support)

Name	zend_runtime_obfuscate
Syntax	bool zend_runtime_obfuscate()
Results	Start runtime-obfuscation support that allows limited mixing of obfuscated and un-obfuscated code.

FAQ

This section contains Frequently Asked Questions and answers regarding encoding and encoded files. Additional FAQ and Support can be obtained at the Zend Website (www.zend.com).

Contents:

- Can encoded and non-encoded PHP files be used together?
- Will using encoded files (instead of source files) change run-time speed or file size?
- When I try to run an encoded file, error messages are displayed. Why?
- Can Zend Intermediate Code files be de-coded back into the PHP source file?
- Do my clients need to install anything to run Zend-encoded files?
- What if my script is dependent on library file?
- My scripts use "include_path" or "auto_prepend". Can I encode them without modification?
- How do I check if a file is encoded?
- Is it possible to generate licenses for an encoded project on a different machine than the machine the project was encoded with?
- Is it possible to use two systems one to encode and the other to generate licenses (for the project encoded on the separate machine)?
- Why are certain files are not being encoded?
- What Optimizer and PHP version should I use?

Can encoded and non-encoded PHP files be used together?

Yes. Encoded and non-encoded PHP files, in most instances, can be used together transparently. The only exception is files which were encoded using the "Work Exclusively with Encoded Files" option. These files will only work with encoded files that were encoded by the same company.

Obfuscated files however, can have problem running together with non-obfuscated scripts, if functions or classes defined in one script are called from another script. Such scripts should be both obfuscated or both non-obfuscated, or classes or functions that are called across the "obfuscation border" should be exported.

To temporarily replace obfuscated script with non-obfuscated one and you still want to call the functions and classes in the script, you can use

zend_runtime_obfuscate() function - however this function may impact performance so it is not recommended as a permanent solution.

Will using encoded files (instead of source files) change run-time speed or file size?

There might be some speed gain because the compilation stage is saved on every run of the script, however, speed improvements, if any, are dependent on the nature of the script.

The size of an encoded file might be somewhat smaller or larger than the source file, but this too is dependent on the nature of the script. The factors that tend to improve run-time speed are not necessarily the same as those that tend to decrease file size.

When I try to run an encoded file, error messages are displayed. Why?

The most common cause of error messages and failure is incompatibility, either with the PHP version, or with the Zend Intermediate Code file (that is, with the version of the Zend Guard that encoded the file). For additional information, see the "Avoiding Incompatibilities" section in the Zend Optimizer User Guide.

Can Zend Intermediate Code files be de-coded back into the PHP source file?

Obfuscation like encryption can only be decoded using brut-force techniques. Such techniques typically require vast amounts of time and resources to decode obfuscated strings. Furthermore, the longer the obfuscated string is the less realistic it is to decode these strings as the number of possible combinations increases exponentially. Therefore, it is not feasible that Obfuscated files will not be successfully de-coded.

Do my clients need to install anything to run Zend-encoded files?

Yes, they need to install the Zend Optimizer as part of their PHP setup. The Zend Optimizer is available as a freeware download from:

http://www.zend.com/products/zend_optimizer.

What if my script is dependent on library file?

Files that are include()'d or require()'d in the script must be present at run time and will not be part of the encoded file. Both the encoded file and the library files should be shipped together. Such files can be either encoded or non-encoded – both file types can be used together transparently.

Note:

If "Work Exclusively with Encoded Files" is used the library files must also be encoded by the same company.

Keeping certain files as plain text can be useful if you wish to let users customize parts of the application.

However when obfuscating files there may be a problem running non-obfuscated scripts, if functions or classes defined in one script are called from another script. Such scripts should be both obfuscated or both non-obfuscated, or classes or functions that are called across the "obfuscation border" should be exported. To temporarily replace obfuscated script with non-obfuscated one and you still want to call the functions and classes in the script, you can use `zend_runtime_obfuscate()` function - however this function may impact performance so it is not recommended as a permanent solution.

My scripts use `include_path` or `auto_prepend`. Can I encode them without modification?

Yes. Since each file is encoded separately, all `include()`'s take place at run-time and therefore, do not interfere with the Zend Guard.

How do I check if a file is encoded?

You can use a PHP script with the API: `boolean zend_loader_file_encoded()` to check if the file is encoded.

Zend encoded files return = `true`.

Is it possible to generate licenses for an encoded project on a different machine than the machine the project was encoded with?

Yes, as long as the Guard license is good for both machines, since the license generator would require a license to run.

Is it possible to use two systems one to encode and the other to generate licenses (for the project encoded on the separate machine)?

Yes, if both Host IDs are allowed in the generated license

Why are certain files are not being encoded?

It could be that your file has the same name as a general pattern in the ignore list. To solve this select the file and in the General settings tab click "Override Project Configuration. This will remove the resource from the list of patterns to ignore. Repeat if other files fall under the same category of the general pattern for example; if the pattern to ignore is `'test*.php'` the file `'contest.php'` will also be ignored. Alternatively, refine the patterns you want to ignore in the exclude tab.

What Optimizer and PHP version should I use?

Make sure to use the correct Optimizer and PHP versions as follows:

- Zend Guard 5.0 with PHP 5.X requires Zend Optimizer 3.3.x and above

- Zend Guard 5.0 with PHP 4.X requires Zend Optimizer 2.6+ for encoding only or Zend Optimizer 3.0+ for obfuscation support
- PHP Compatibility: supported PHP versions: 4.2.x through 5.2.x

Index

A

Additional Information	64
Allowed IP.....	64
Attribute	64

C

Can encoded and non-encoded PHP files be used together?.....	97
Concurrent Users	64
Cookie Support	64
Cookies	64

I

IP	64
----------	----

L

Lease Timeout	64
License	64
Limit	64

Limit Exceeded	64
Limit Number of Concurrent	64
Limit to the Following IPs	64
Lock to Zend Host IDs.....	64

P

Product Name	64
--------------------	----

R

Registered To	64
Require Cookies	64

Z

Zend Guard API	64
Zend Guard License	64
Zend Guard Toolbar	64
Zend Host ID	64
Zend_loader_file_license	64