sourceguardian

Version 7.1 for PHP
User Manual

Introduction

An introduction to SourceGuardian 7.1 for PHP

by Team SourceGuardian

This SourceGuardian 7.1 for PHP User Manual covers all of the features in this new exciting version. We hope that you enjoy using our product and find this user guide to be informative.

If there is anything that you feel has been omitted from this user manual, then please let us know as we are passionate about providing excellent service.

Have fun using your new product...

SourceGuardian 7.0 for PHP

Table of Contents

Part I

1 Introduction

1.1 About SourceGuardian for PHP

The SourceGuardian™ products have been built as a suite of professional systems for source code protection. Our team of programmers have created proprietary methods for encrypting code whilst keeping the maximum flexibility for the distribution of your scripts.

Our first product, SourceGuardian™ for PHP was launched in 2002 and quickly rose to become the professionals for PHP code protection. Thanks to our early market entry and the customers who put their trust in us, we've been able to develop SourceGuardian™ into a leading protection solution used by thousands across the world.

The most exciting thing about SourceGuardian™ for us is how we constantly hear from our clients how SourceGuardian™ has finally enabled them to distribute their commercial code and how developers are able to solve many of the problems that plague them when coding for a specific client. We hope to enable many more!

As for the future of SourceGuardian™, our PHP product has really taken us aback with the huge uptake and acceptance in the market and we thank everyone who has purchased, downloaded or even taken the time to browse our site. We plan to continue to increase the functionality and power of these programs whilst keeping an affordable upgrade path.

Thanks for your interest, and thanks for your business.

The SourceGuardian™ Team

1.2 How to buy

To purchase SourceGuardian™ 7.1 for PHP, please visit the following:

Website: http://www.sourceguardian.com/purchase/index.php

There are two methods available: via credit card or via Paypal.

1.3 Features

SourceGuardian™ 7.1 for PHP Features List

Protection method

The SourceGuardian™ 7.1 for PHP Encoder protects PHP scripts by compiling PHP source code into a bytecode format and this is followed by encryption. This protects your scripts from reverse engineering.

Supported PHP versions

SourceGuardian™ 7.1 for PHP works with the following versions and above: PHP 4.1.x, 4.2.x, 4.3.x, 4.4.x and PHP 5.0.0 and higher are fully supported.

Interface

A ground-up redevelopment of SourceGuardian™, including a new reworked GUI is now available. The Windows version has both Wizard and Advanced modes, giving you powerful tools and features to protect your code. In addition, we have also developed a powerful cross-platform command line encoder that runs under Windows and Linux.

Locking

To protect your scripts from unauthorised usage SourceGuardian™ 7.1 for PHP has added features that can optionally lock your scripts to run only from predefined IP addresses, domain names or LAN hardware addresses (MAC). SourceGuardian™ 7.1 for PHP can also easily produce trial versions of your scripts by setting an expiry date for the script or by limiting the number of days that protected script will work. For larger projects SourceGuardian™ 7.1 for PHP provides an option to protect an entire project so that all scripts used in the project will work only with other protected scripts. No one may include a protected script from another unprotected script and this adds another level of protection.

Here is a sample list of features:

- locking to date
- locking to multiple domain names
- locking to multiple ip addresses
- locking to multiple LAN hardware (MAC) addresses

- improved locking to a specific domain name with encryption. The domain name is used as a part of key for encryption, so protected scripts may not be decrypted and run from another domain.
- improved locking to the ip address with encryption. The ip address is used as a part of key for encryption. This means that protected scripts cannot be decrypted and run from another ip address.
- locking of an entire PHP project, so that no protected script can run if any other script is substituted with an unencoded one or encoded with another installation of SourceGuardian™ . This is ideal for protecting settings, passwords etc within a PHP project.
- A new feature is the ability to lock with an external license file produced by the SourceGuardian™ 7.1 for PHP license generator. This is Ideal for creating protected scripts to be distributed between different users and it will even allow different options for different users. The SourceGuardian™ 7.1 for PHP license generator tool can run under Window and Linux as command line tool which adds another powerful element - It provides a method for licenses to be dynamically generated and this would be useful (for example) when selling scripts online.

Other options

The following is not an exhaustive list, but covers some of the other options in this new version:

- automatic backup of source files
- multiple files processing: enumerated, file mask optionally with directory recursion or file list from the command line
- encoding confirmation when run from the command line
- option to include HTML or PHP code to run before a protected script. This is best for including copyrights or for any other advanced needs
- option to replace standard error handler when the appropriate loader is not found. Both HTML or PHP code can be included here
- option to allow asp-style tags in source files
- option to allow short PHP's tags in source files
- option to exclude the automatic script loader from protected scripts for advanced users and manual loader installation

Cross platform

Cross platform encoding. A script encoded under one operating system will run under any other supported operating systems. Currently we have an encoder for Windows and Linux and Script Loaders will run under Windows, Linux and FreeBSD. In the near future we will support more operating systems.

Thread Safety support

SourceGuardian™ 7.1 for PHP has a special versions of its Script Loader for Thread Safety PHP installations under Windows and Linux.

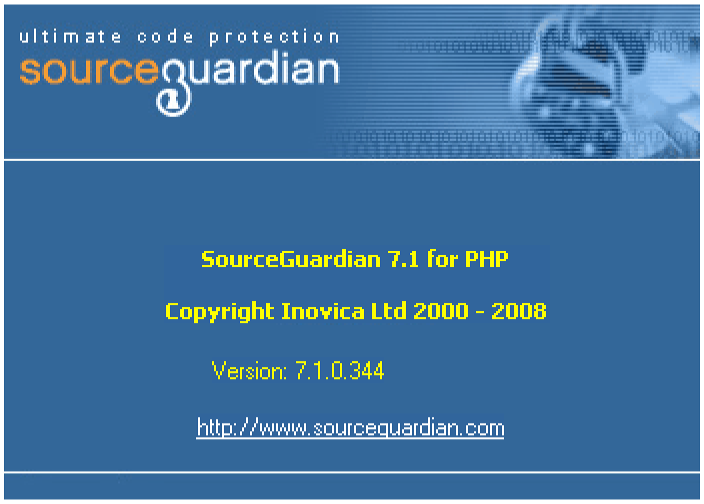
Evaluation

We provide a Free 7 days evaluation of SourceGuardian™ 7.1 for PHP

Part II

2 GUI manual

2.1 Overview



SourceGuardian 7.1 for PHP

Copyright Inovica Ltd 2000 - 2008

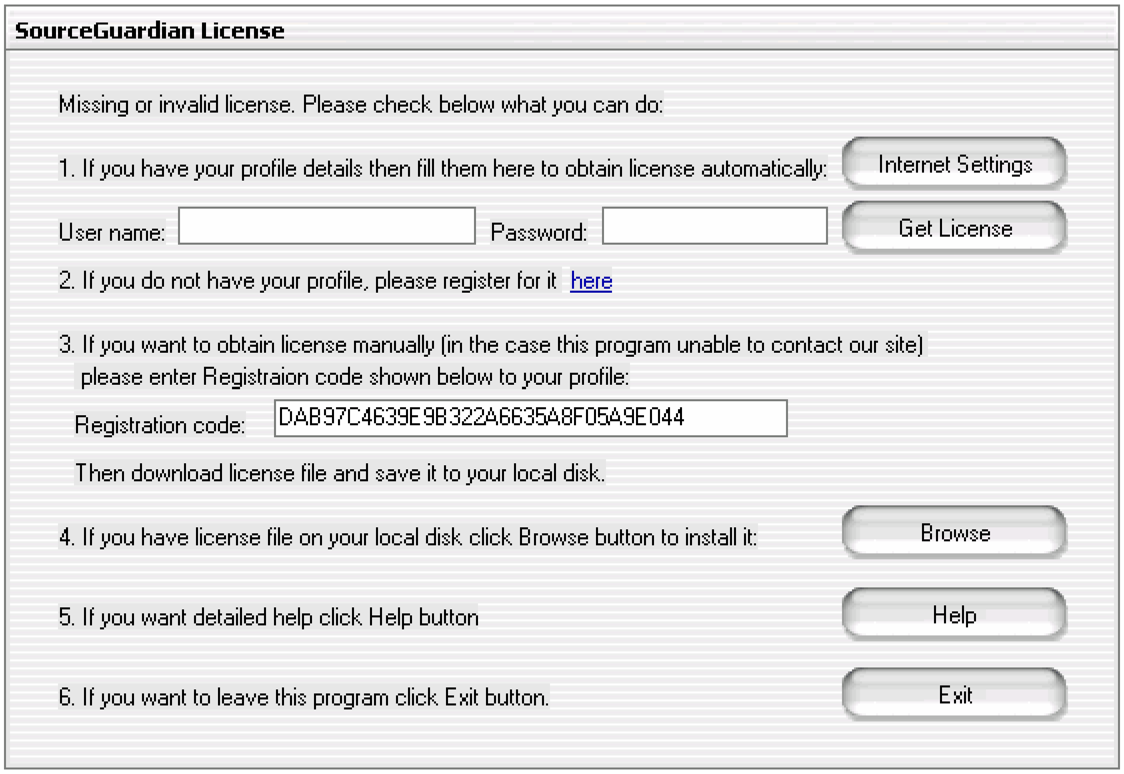
Version: 7.1.0.344

http://www.sourceguardian.com

The SourceGuardian™ 7.1 for PHP GUI is a Windows-based, user-friendly graphic interface to our command line encoder. It uses all the powerful features that command line encoder offers, while adding many additional useful enhancements to the encoding routine. The GUI has an interface which is skinnable (27 skins included) and you can select the skin that you like the most. It has a Wizard mode for new users who want to make quick start in using the GUI and there is also an Advanced mode where the experienced user can use of the power of our encoder.

2.2 First step - obtaining license

On your first of run of SourceGuardian™ 7.1 for PHP you should see following screen:


SourceGuardian License

Missing or invalid license. Please check below what you can do:

1. If you have your profile details then fill them here to obtain license automatically:

User name: Password:

2. If you do not have your profile, please register for it here

3. If you want to obtain license manually (in the case this program unable to contact our site) please enter Registraion code shown below to your profile:

Registration code: DAB97C4639E9B322A6635A8F05A9E044

Then download license file and save it to your local disk.

4. If you have license file on your local disk click Browse button to install it:

5. If you want detailed help click Help button

6. If you want to leave this program click Exit button.

This screen means that you need to obtain a license first in order to run SourceGuardian™ . In general you can do this via two different ways.

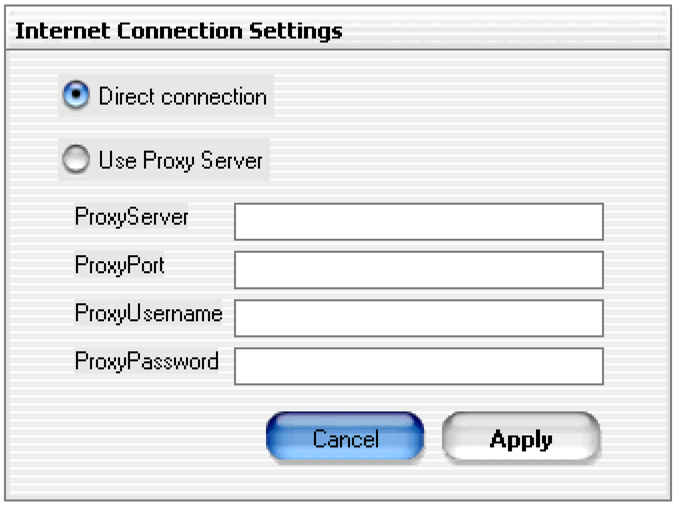
1. Obtaining a license from the application itself.

This is the fastest way to obtain a license, but to do it you need a direct connection to the internet or configure a Proxy server so that the SourceGuardian™ 7.1 application can connect to the internet.

NOTE: Some firewalls may prohibit SourceGuardian™ from connecting to the internet, so you may have to enable internet access for SourceGuardian™ or obtain your license by another method (see 2.). On how to enable internet access for a custom application with your firewall please consult your firewall documentation.

When you purchase the full version of SourceGuardian™ 7.1 for PHP, or request demo version of it, you will receive an email with details on how to access your profile on our site. This email contains a user name and a profile password. Just type them into the 'User name' and 'Password' fields and click on the 'Get License' button. After the license has been downloaded you will see a message box saying: "Application will now close. Please start the application again to activate your copy". Press 'Ok' and then start the program again. If everything has installed correct, the application will start normally. If anything has gone wrong with the installation, you will see the 'License error' screen again. Make sure you enter your user name and password correctly, check your internet settings and try again. If this still will not allow access please try a different method (see 2.).

Configuring internet connection. This is useful if you normally need to access the internet via a proxy server. Click on the 'Internet Settings' button and you will see following window:



Direct connection - This is selected by default and is used when you have a direct internet connection. Use Proxy Server - Select this option if you are behind a Proxy Server. Enter the Proxy Server address and the Port. Also the Proxy Username and Password should be entered if it is applicable. Please consult your network administrator about information about your Proxy Server.

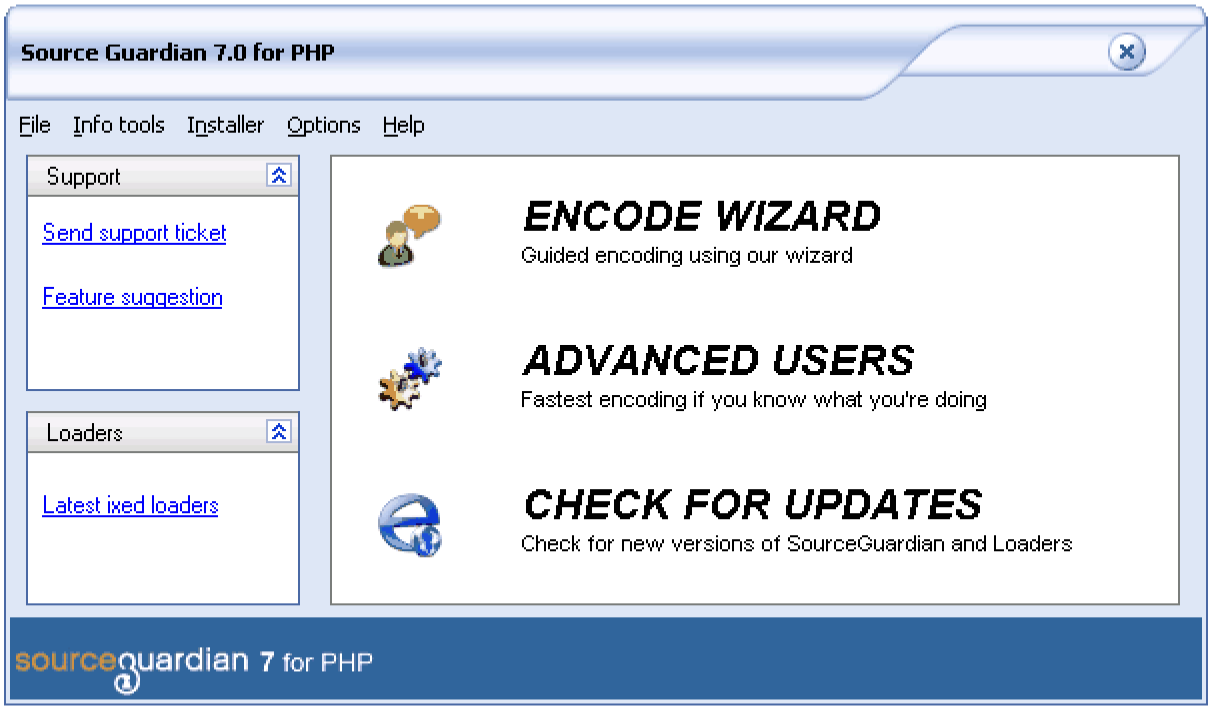
2. Obtaining a license via the "your profile" section on our site.

When you are unable to obtain license via the normally method, you can use this method to retrieve and download a license file. Go to the profile login page on the SourceGuardian.com website. Type your user name and password.

Once you have entered the system select registration code from within the SourceGuardian™ 7.1 for PHP application (by double clicking on it). Copy and paste it to the corresponding field of the "your profile" area on our site ("Please enter your registration key here to generate license:").

When you have done the above, Click on 'Submit' and this will generate license. To download it click on the 'Download' link in the 'Available licenses' section. Save this license, somewhere on your PC. Then in the SourceGuardian™ 7.1 for PHP application click on the 'Browse' button, select the license file you just downloaded and saved. You should now see a message box saying 'Application will now close. Please start SourceGuardian™ 7.1 for PHP again to activate your license!'. Click Ok. And start SourceGuardian™ 7.1 for PHP again.

2.3 Startup screen



The following outlines the features available on this screen:

Click on the Encode Wizard band if you want use our wizard to help you with Encoding. This is recommended for beginners.

Click on the Advanced Users band to switch to Advanced mode, where you can control all of the settings and parameters. This is recommended for advanced users.

Click on Check for Updates. Currently it opens up your profile where you can download latest version of the software.

On the left-hand side there are the following useful links:

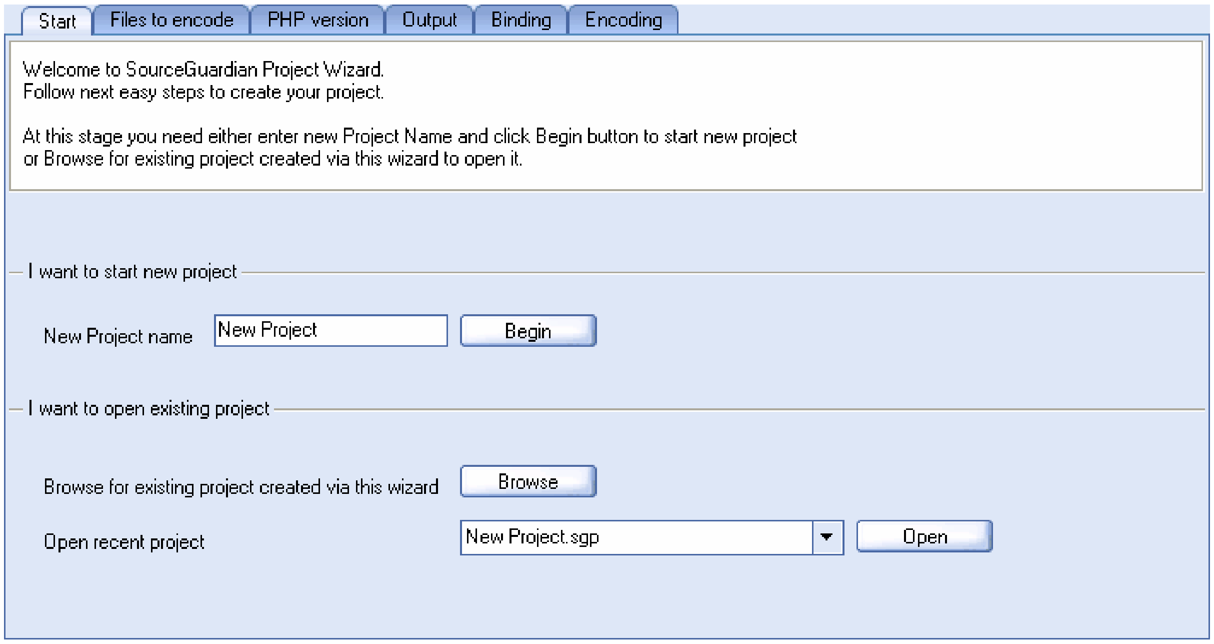
Send support ticket - This opens a page on our site to send support ticket.

Feature suggestion - This opens a page on our site where you can submit your suggestion.

Latest ixed loaders - This will link to our site where you can download the latest ixeds.

2.4 Encode Wizard

2.4.1 Step1 - Project definition



Welcome to SourceGuardian Project Wizard.
Follow next easy steps to create your project.

At this stage you need either enter new Project Name and click Begin button to start new project
or Browse for existing project created via this wizard to open it.

I want to start new project

New Project name: New Project

I want to open existing project

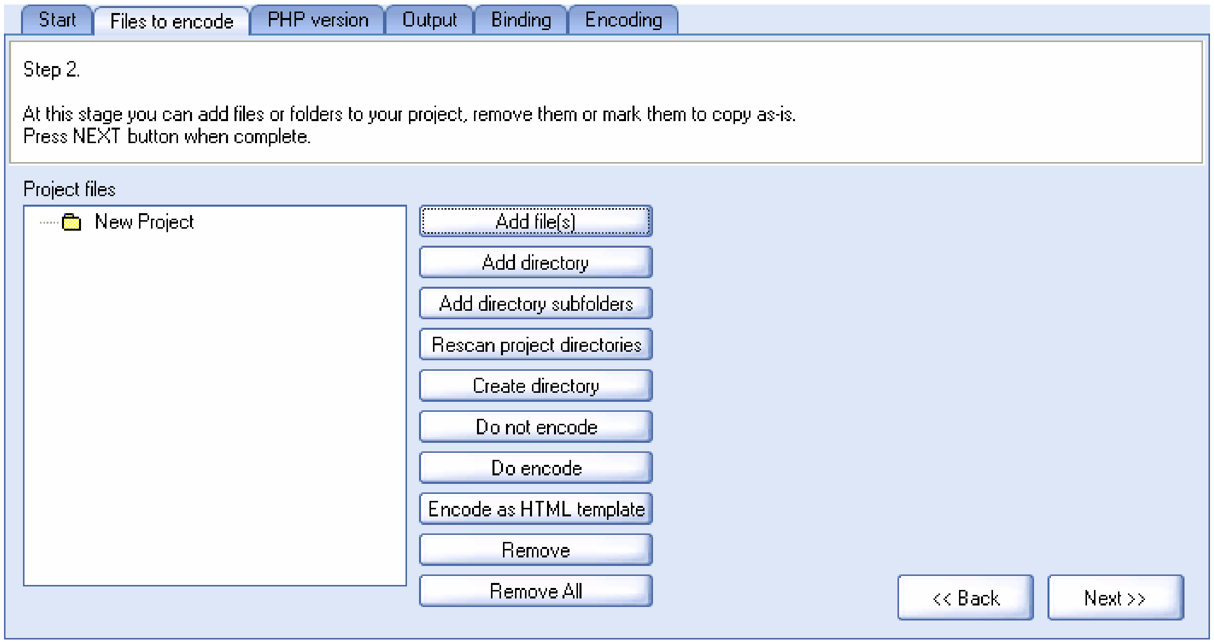
Browse for existing project created via this wizard

Open recent project: New Project.sgp

This is the Start screen. You either need to create a new project or select an existing project.

To create a new project, enter the project name in the "New Project Name" field and click on "begin". If you have an existing project please click on the "Browse" button to find and select your project. Also you can open recent project that you were working on via selecting 'Open recent project' drop down list and click on 'Open' button.

2.4.2 Step2 - Files to encode



Step 2.

At this stage you can add files or folders to your project, remove them or mark them to copy as-is.
Press NEXT button when complete.

Project files

New Project

To add one or more files click on the 'Add file(s)' button. The File selection dialogue appears. Select one or more files to add to your project.

To add a whole directory (with all subdirectories) click on 'Add directory' button and then select the directory you want to add.

To add only subdirectories of a directory click on 'Add directory sub folders' button and then select the directory.

Files/Directories will be added into currently selected directory.

If you want to create new empty directory click on 'Add directory' button.

If you want to mark any given file or directory (with all subdirectories and files) to not encode, then first select the file or directory and click on the 'Do not encode' button. The Icon at the left of the file or directory will change to a red X image (which means that it wont be encoded and will be copied as-is).

If you wish to mark any given file or directory (with all subdirectories and files) to encode then first select the file or directory and then click on 'Do encode' button. The Icon at the left of the file or directory will change to a blank page image (which means that it will be encoded).

If you wish to mark any given file or directory (with all subdirectories and files) to encode as HTML template then first select the file or directory and then click on 'Encode as HTML template' button. The Icon at the left of the file or directory will change to a lamp image (which means that it will be encoded as HTML template).

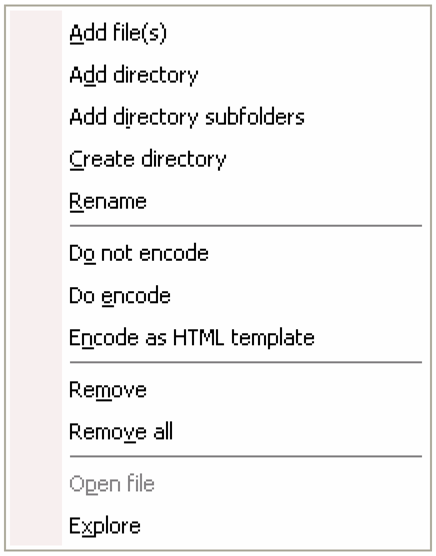
If you wish to remove a file or directory from your project, then select the file or directory and then click on 'Remove' button.

If you wish to remove all files and directories from your project (clear all) then click on the 'Remove All' button.

Since 4.2 you can select multiple files/directories and apply all operations on selected items (Do encode/Do not encode/Remove).

Also you can drag and drop items as you do it usually in Windows Explorer.

If you click right mouse button on tree view popup menu will be shown.



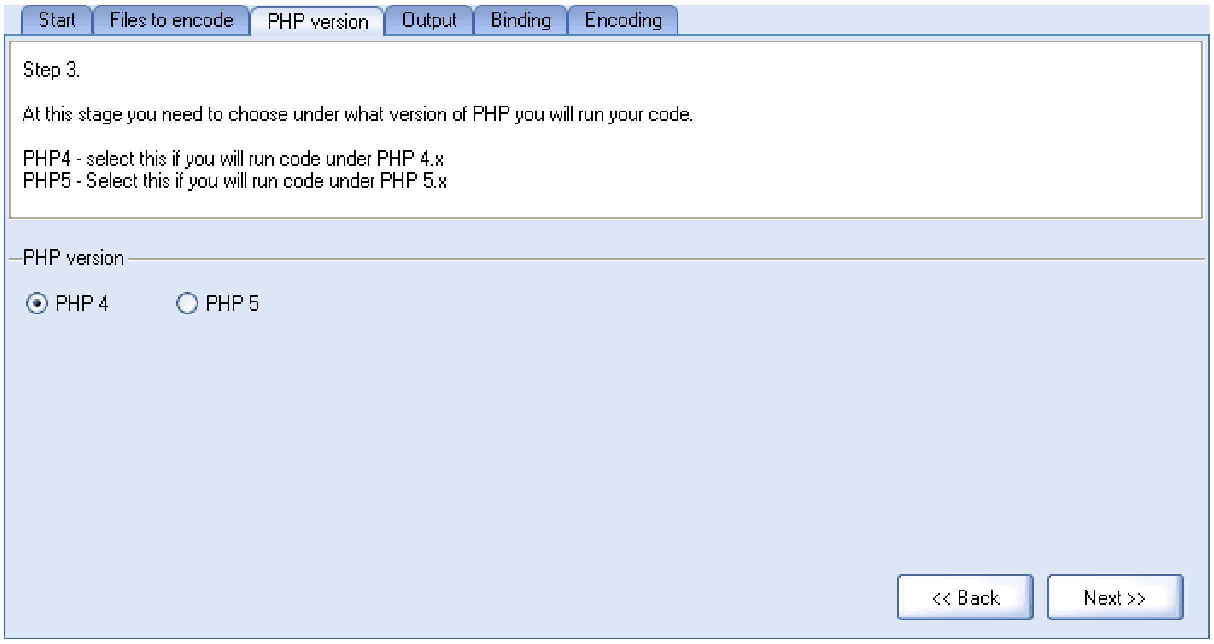
Using it you can do the same actions as described above plus 'Rename', 'Explore' and 'Open file'. 'Rename' allows you to change directory name. 'Open file' opens file using application associated with it (for example your PHP editor).
'Explore' - opens currently selected directory in explorer

When you have finished click on the 'Next>>' button.

How to change the name of your project:

If you wish to change the name of your project, select the root of the files tree (this is the name of your current project). Then click on it once more and wait. This text string then become editable. Change it to whatever you want.

2.4.3 Step3 - Selecting PHP version



Step 3.

At this stage you need to choose under what version of PHP you will run your code.

PHP4 - select this if you will run code under PHP 4.x
PHP5 - Select this if you will run code under PHP 5.x

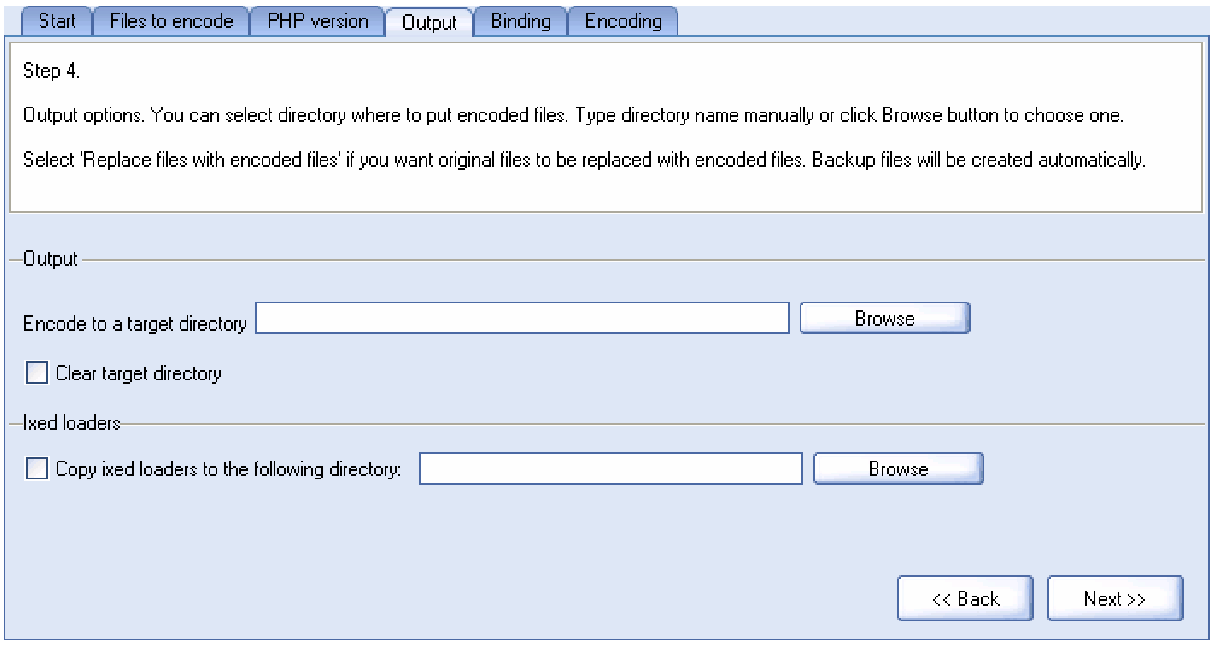
PHP version

[✓ PHP 4] PHP 5

On this screen you can choose between PHP 4.x.x or PHP 5.x.x encoding. Your choice should be easy when you know what version of PHP is installed on the server where you plan to run your scripts. You can also create two different versions of your scripts if you have clients who may have a preference for a particular version of PHP.

NOTE: Scripts encoded for PHP 4.x.x will not be able to run under PHP 5.x.x and vice versa.

2.4.4 Step4 - Output options



Step 4.

Output options. You can select directory where to put encoded files. Type directory name manually or click Browse button to choose one.

Select 'Replace files with encoded files' if you want original files to be replaced with encoded files. Backup files will be created automatically.

Output
Encode to a target directory
Browse
Clear target directory
Ixed loaders
Copy ixed loaders to the following directory:
Browse
<< Back
Next >>

<u>Output</u>

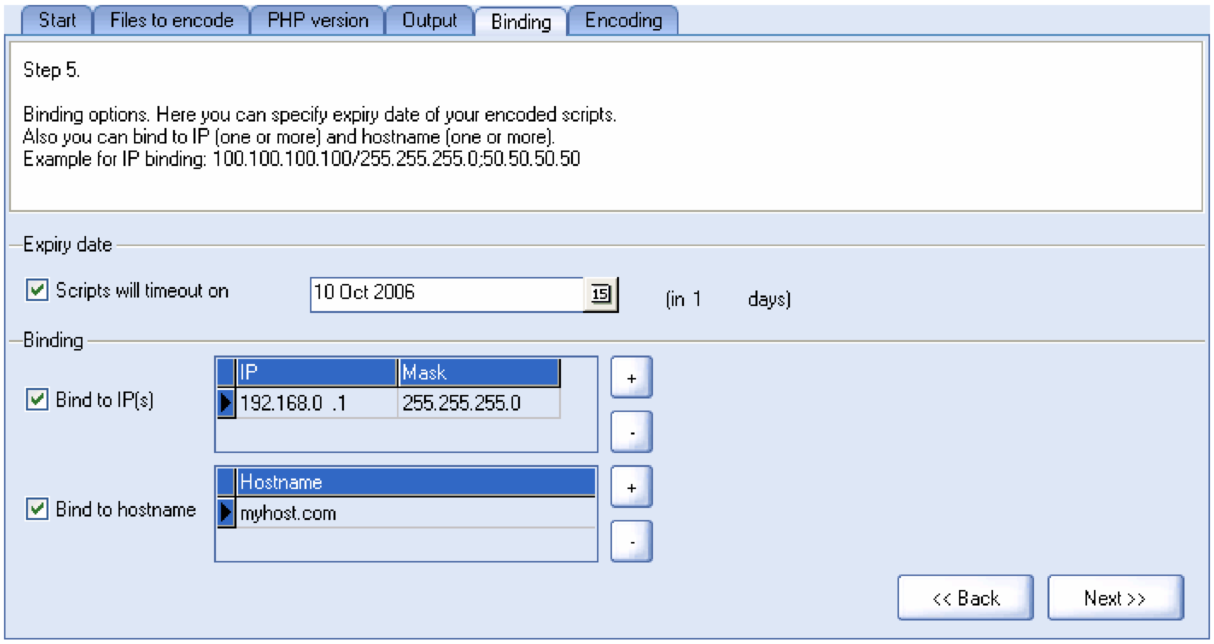
'Encode to a target directory' - click on browse and select an existing directory or type the full directory name manually. If the directory does not exist it will be created automatically. Since 4.2 only this method of output is allowed.

'Clear target directory' - select this if you want to clear all files/directories from the target directory. **WARNING!!!** All files and directories removed permanently - you cannot restore them via recycle bin.

<u>Ixed loaders</u>

'Copy ixed loaders' - This allows you to specify the directory where you want to copy the ixed loaders after encoding process has completed. This field is filled in automatically when you change the 'Encode to a target directory' field value. You can change it to any other path if you wish. If the directory does not exist it will be automatically created.

2.4.5 Step5 - Binding options



Step 5.

Binding options. Here you can specify expiry date of your encoded scripts.
Also you can bind to IP (one or more) and hostname (one or more).
Example for IP binding: 100.100.100.100/255.255.255.0;50.50.50.50

Expiry date

☑ Scripts will timeout on 10 Oct 2006 (in 1 days)

Binding

☑ Bind to IP(s)

IP	Mask
192.168.0 .1	255.255.255.0

☑ Bind to hostname

Hostname
myhost.com

Expiry date

'Scripts will timeout on' - click on calendar icon to choose the date when you want to set an expiration date for the script. At the left side you will see the hint showing how many days are left from today's date.
The script will not run on and after the specified date and comes with the error message: "script has expired".

Binding

'Bind to IP(s)' - bind script to an ip/mask. The encoder will lock the script to run only from the specified IP address(es). The specified IP address mask will be applied to the real IP address before comparing. So you may use this option to lock the script to a multiple IP if mask is specified. If run from not allowed IP script will come with the error message: "script cannot run on this machine" You may add as many IP address/mask pairs as you want using editor. Press '+' button if you need to add another IP/Mask pair (or press down arrow on keyboard while in editor). Press '-' button if you want to delete current IP/Mask pair.

IP address mask 255.255.255.255 is used by default if not specified.

Bind to hostname: - you can bind the script to a domain name. The Encoder will lock the script to run only from the specified domain and all sub domains. If an attempt is made to run the script on a non-authorised domain, the following error message will be displayed: "script cannot run on this machine". You may add as many domain names as you want.

Hint: use the name of the main domain in this option, not the name of any sub domain until you are sure you need to lock to a sub domain.

Example 1: mydomain.com

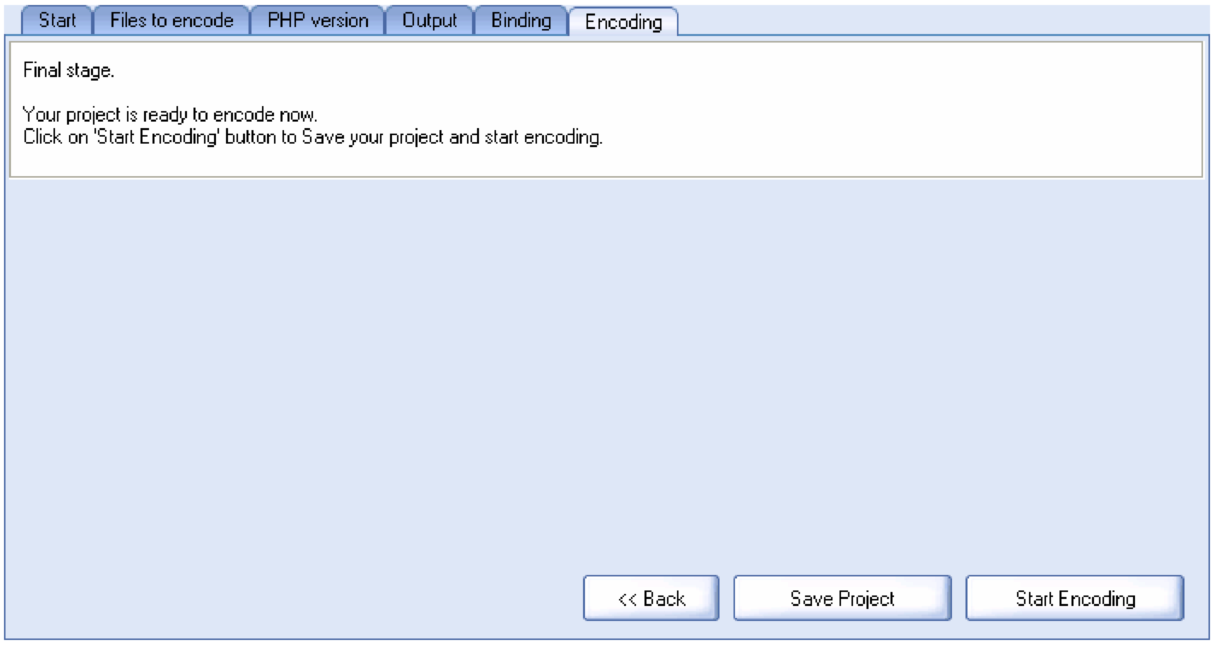
The script will run from mydomain.com, www.mydomain.com, myname.mydomain.com etc but will

NOT run from otherdomain.com, www.otherdomain.com, otherdomain.net etc.

Example 2: www.mydomain.com

Script will run ONLY from www.mydomain.com. It will not run on the Main domain mydomain.com and all other subdomains like myname.mydomain.com as well as other domains like otherdomain.com, www.otherdomain.com, otherdomain.net etc.

2.4.6 Step6 - Encoding



Final stage.

Your project is ready to encode now.
Click on 'Start Encoding' button to Save your project and start encoding.

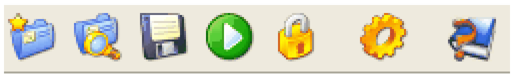
At this stage you may wish to save your project if you may wish to use the same options at a later date. In order to do this, click on the 'Save Project' button. We would recommend doing this before encoding, as it is the last chance to do it.

'Start Encoding' - this actually starts the encoding process. See Encoding process chapter.

2.5 Advanced Users

2.5.1 Toolbar Icons description

The Toolbar shown in the Advanced mode is shown below:



Opens New Project dialogue.

Open existing Project.

Save current Project.

Start Encoding.

License generator.

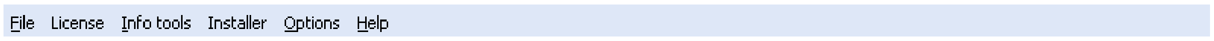
Options dialogue.

Opens this help.

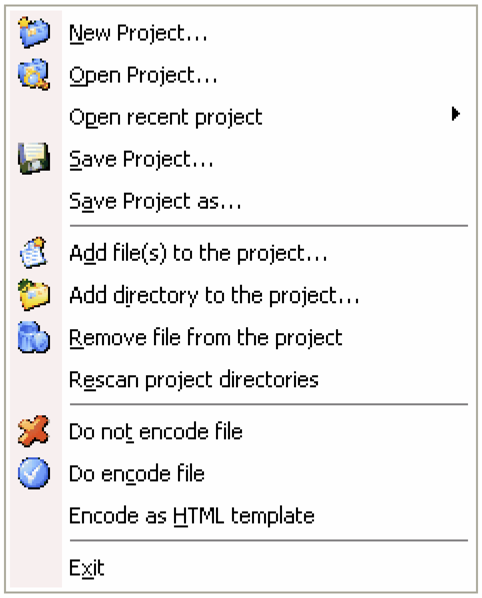
All icons have tool tips if you have them switched ON in Options (ON by default).

2.5.2 Menu items description

Menu bar:



File menu:



New Project - New Project Dialogue.
Open Project - Opens existing Project.
Open recent project - allows to quickly open recently saved/opened project without browsing for it
Save Project - Saves current Project.
Save Project as - Save current Project to a specified file.

Add file(s) to the project - allows the user to add file(s) to the current Project.
Add directory to the project - allows adding a whole directory (with all subdirectories and files) to your current Project.
Remove file from the project - removes the selected file or directory from your project.
Rescan project directories - rescan your project to update for added/removed files automatically

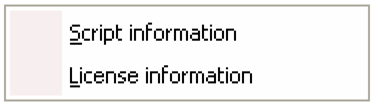
Do not encode file - mark the current selected file or directory to not encode.
Do encode file - mark the current selected file or directory to encode.
Encode as HTML template - mark the current selected file or directory to encode as HTML template

Exit - exits SourceGuardian™ .

License menu:

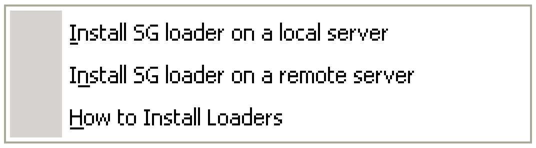
Opens the generate license dialogue.

Info tools menu:



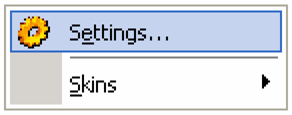
Allows to open information tools - Script information and License information.

Installer menu:

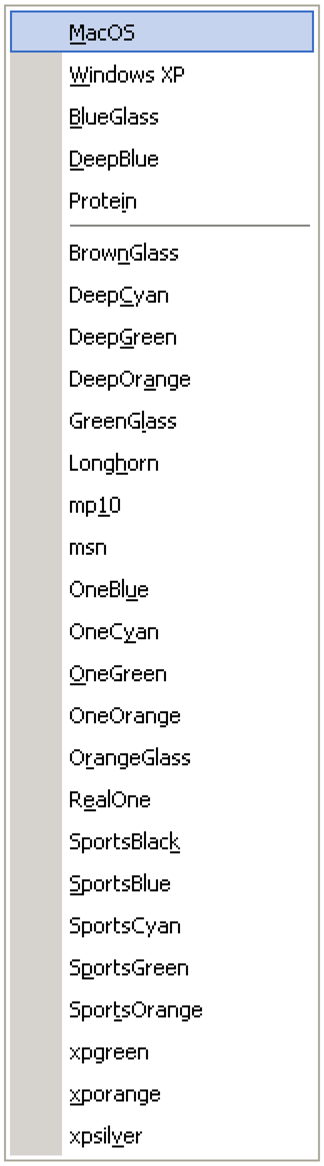

Install SG loader on a local server
Install SG loader on a remote server
How to Install Loaders

Allows to open installers - on local server and remote server.
How to Install Loaders - helps you to know how to install the loader on your destination system.

Options menu:



Settings - opens the settings dialogue.
Skins - skins selection menu:

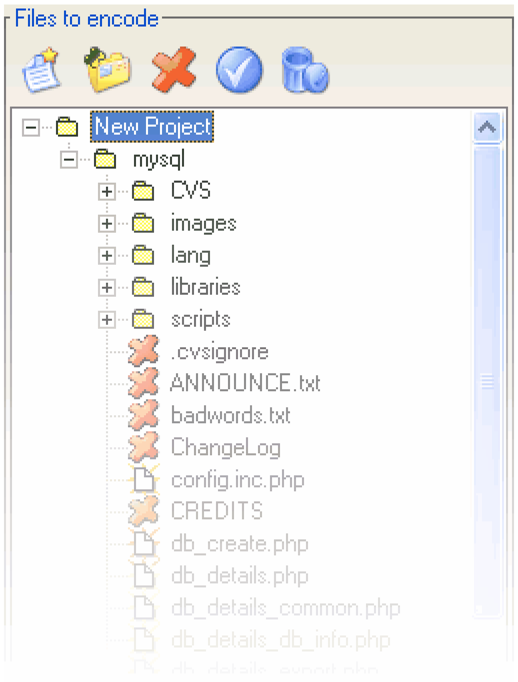


Choose one of the skins listed in this menu. Please wait until the skin has changed before any new actions (several seconds).

Help menu:

From this menu you can open this help and also the About SourceGuardian information.

2.5.3 Adding files to your project



To add one or more files click on the 'Add file(s)' icon.The File selection dialogue appears. Select one or more files to add to your project.

To add a whole directory (with all subdirectories) click on 'Add directory' icon and select the directory you want to add.

If you want to mark any given file or directory (with all subdirectories and files) to not encode, then first select the file or directory and click on the 'Do not encode' icon. The Icon at the left of the file or directory will change to a red X image (which means that it will not be encoded and will be copied as-is).

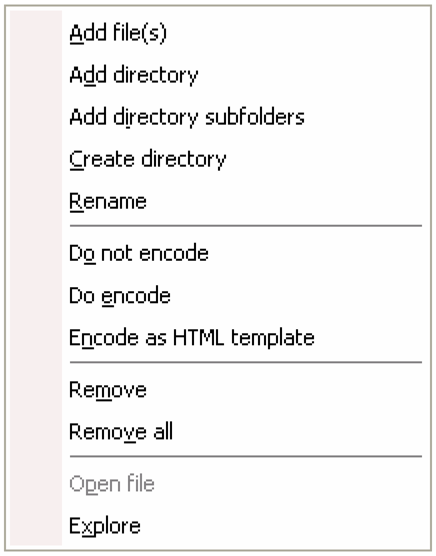
If you want to mark any given file or directory (with all subdirectories and files) to encode then first select the file or directory and then click on the 'Do encode' icon. The Icon on the left of the file or directory will change to a blank page image (which means that it will be encoded).

If you want to remove some file or directory from your project then select the file or directory and then click on the 'Remove' icon.

Since 4.2 you can select multiple files/directories and apply all operations on selected items (Do encode/Do not encode/Remove).

Also you can drag and drop items as you do it usually in Windows Explorer.

If you click right mouse button on treeview popup menu will be shown.

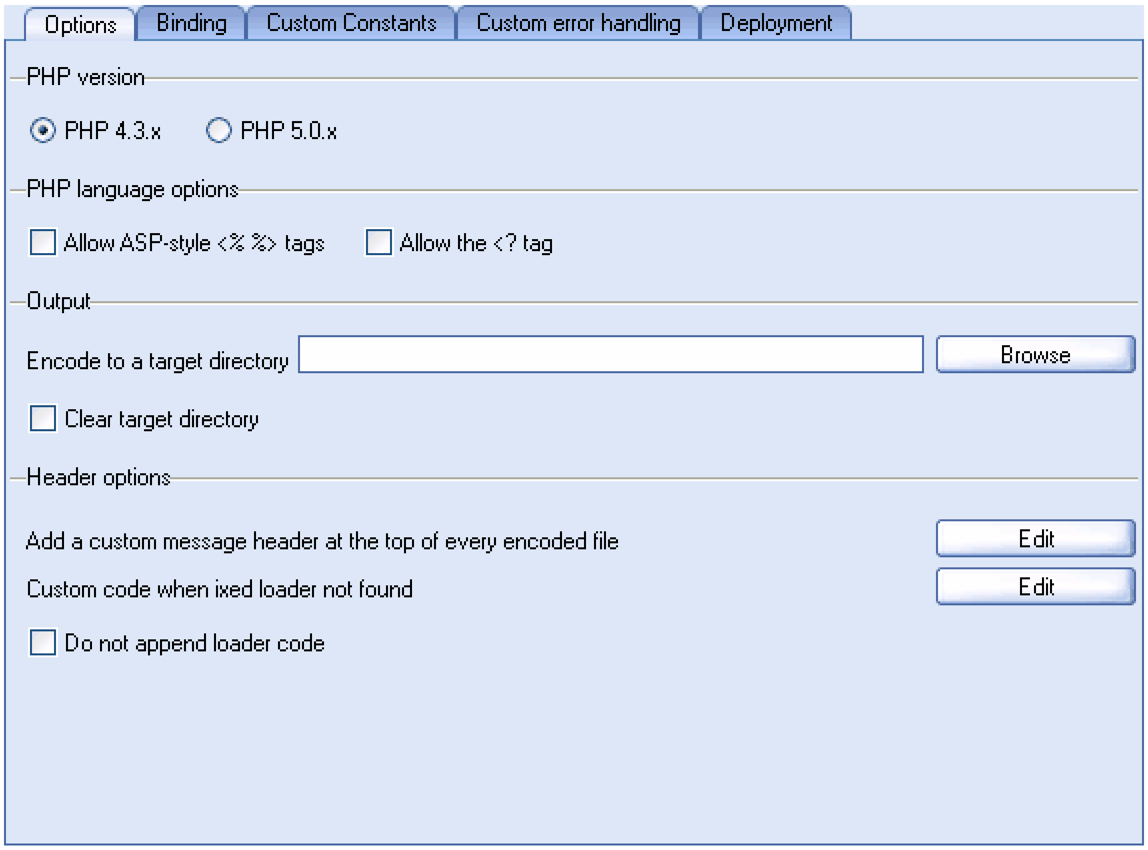


Using it you can do the same actions as described above plus 'Rename', 'Encode as HTML template', 'Explore' and 'Open file'.
'Rename' allows you to change directory name.
'Open file' opens file using application associated with it (for example your PHP editor).
To add only subdirectories of a directory click on 'Add directory subfolders' item and then select the directory.
'Encode as HTML template' - allows you to mark file/directory to encode them as HTML template.
'Explore' - opens currently selected directory in explorer

<u>How to change the name of your project</u>:

Select the root of the files tree (this is the name of your current project). Then click on it once and wait. This string become editable. Change it to whatever you want.

2.5.4 Encoding options



PHP version

(● PHP 4.3.x) ○ PHP 5.0.x

PHP language options

☐ Allow ASP-style <% %> tags ☐ Allow the <? tag

Output

Encode to a target directory

☐ Clear target directory

Header options

Add a custom message header at the top of every encoded file

Custom code when ixed loader not found

☐ Do not append loader code

PHP version:

On this screen you can choose between PHP 4.x.x or PHP 5.x.x encoding. Your choice should be easy when you know what version of PHP is installed on the server where you plan to run your scripts. You can also create two different versions of your scripts if you have clients who may have a preference for a particular version of PHP.

NOTE: Scripts encoded for PHP 4.x.x will not be able to run under PHP 5.x.x and vice versa.

PHP language options:

Allow ASP-style <% %> tags - enables the use of ASP-like <% %> tags in addition to the usual <?php ?> tags.
Allow the <? tag - allow the <? tag, otherwise only <?php and <script> tags are recognized.

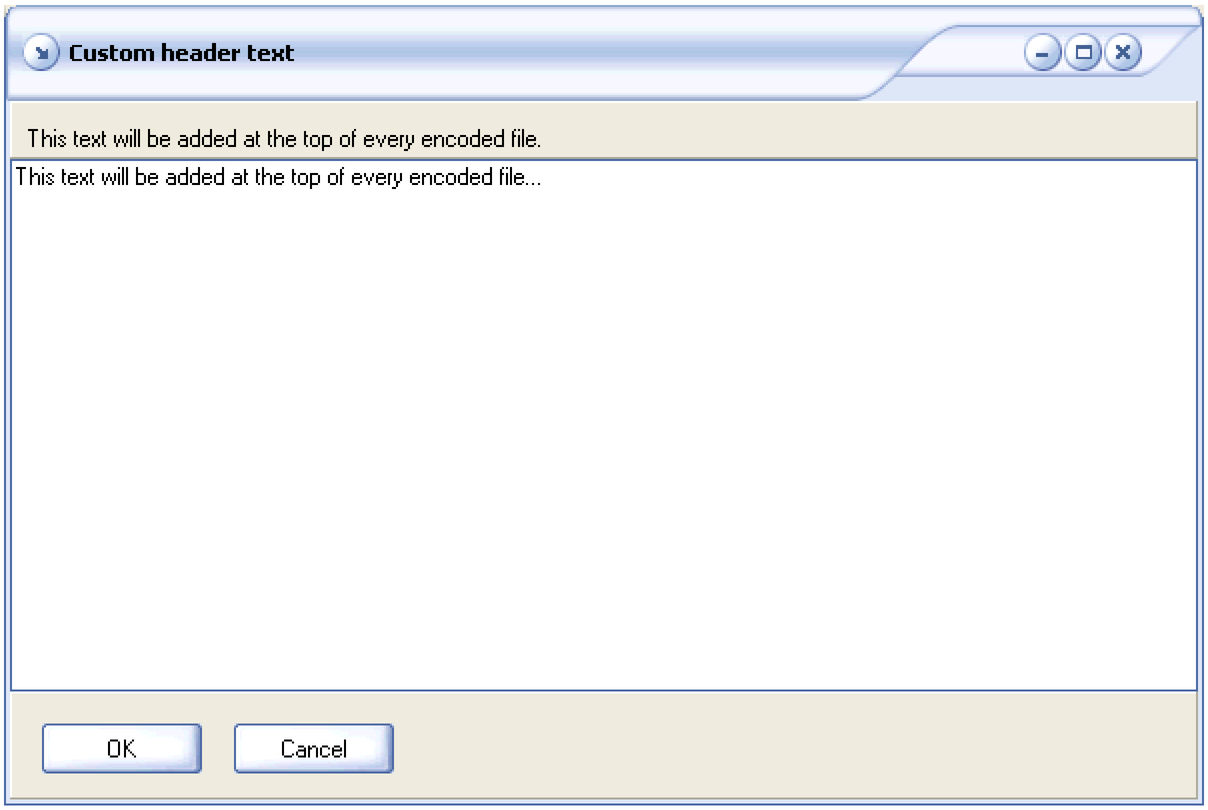
Output:

'Encode to a target directory' - click on browse and select an existing directory or type the full directory name manually. If the directory does not exist it will be created automatically. Since 4.2 only this method of output is allowed.

'Clear target directory' - select this if you want to clear all files/directories from the target directory.
WARNING!!! All files and directories removed permanently - you cannot restore them via recycle bin.

Header options:

Add a custom message header at the top of every encoded file:

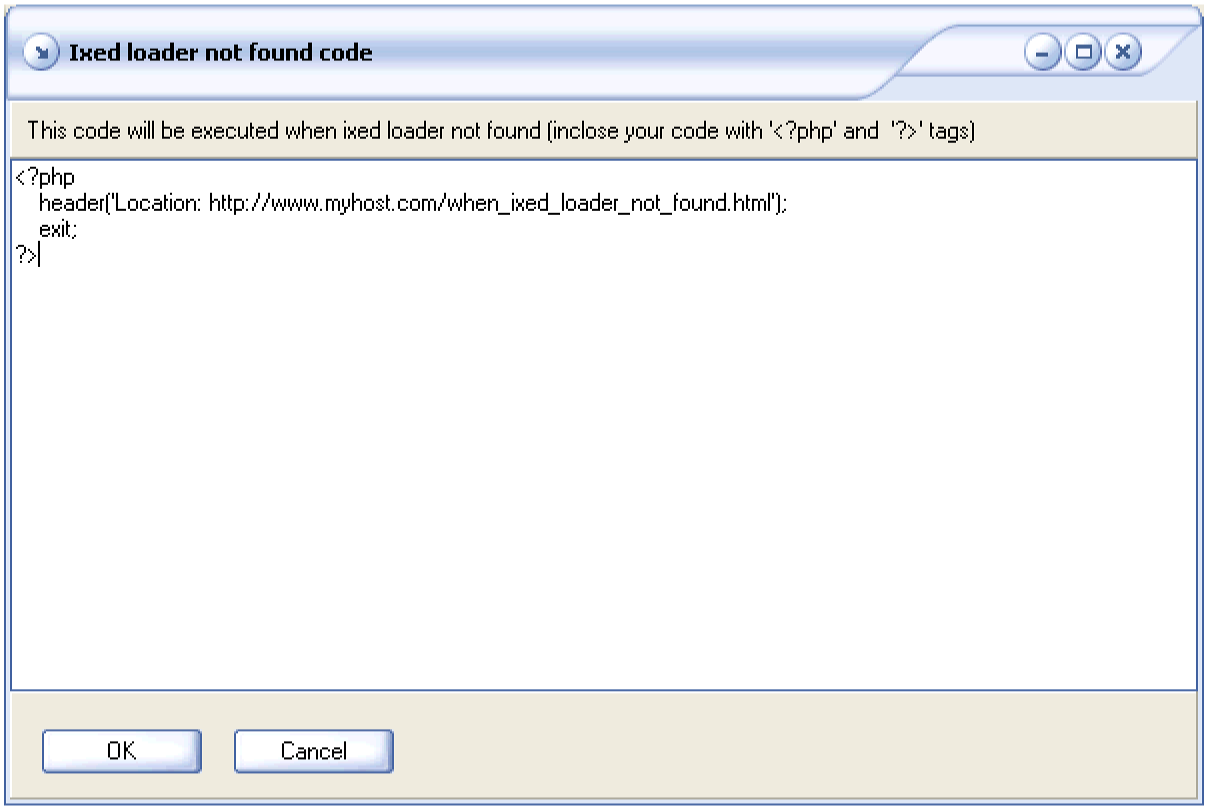


This text will be added at the top of every encoded file.

This text will be added at the top of every encoded file...

Prepend header code. You may put any code to be executed BEFORE the protected scripts code. This code WILL NOT BE ENCODED. This may be either HTML text or PHP code. For PHP code - you should use <?php ?> tags. This option is usually used for including copyrights into protected scripts.

Custom code when ixed loader not found:



This code will be executed when ixed loader not found (inclose your code with '<?php' and '?>' tags)

```
<?php
   header('Location: http://www.myhost.com/when_ixed_loader_not_found.html');
   exit;
?>
```

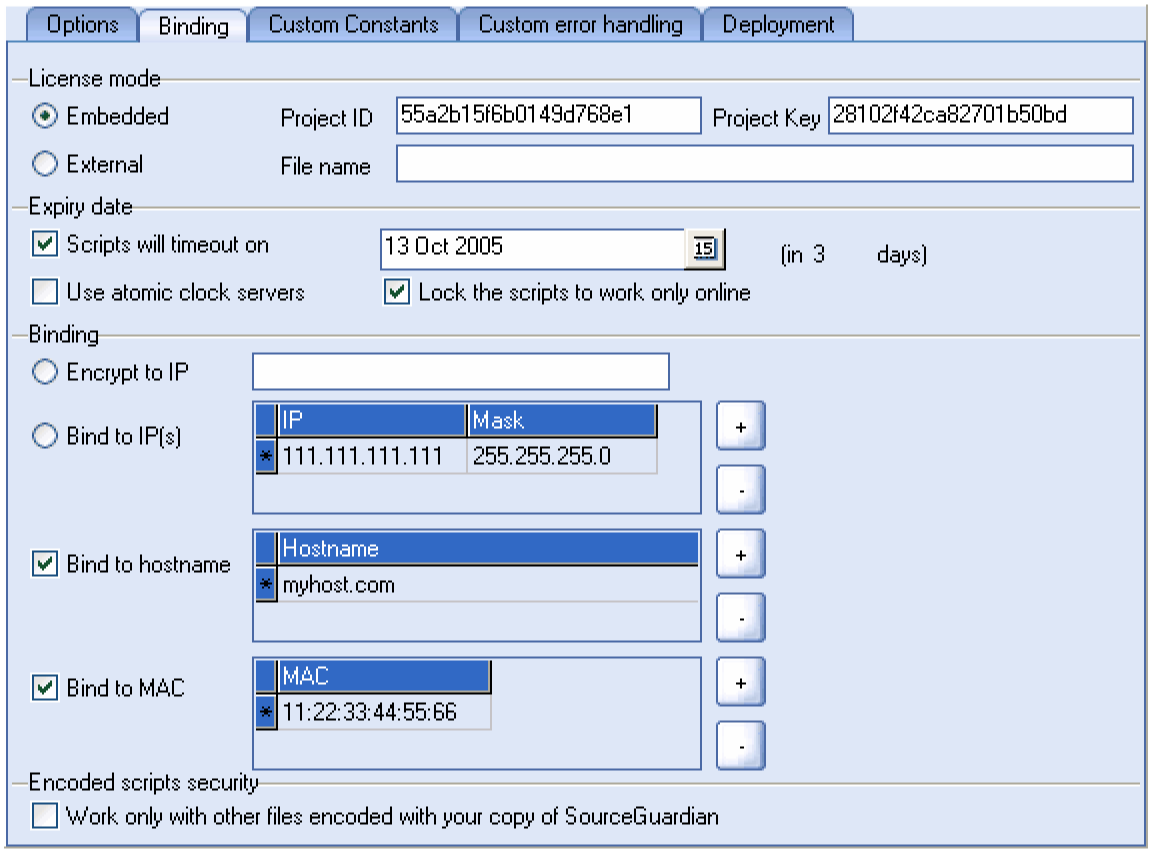
It is possible for you to change the default loader error code. This option allows you to change the default error action of the protected script if it cannot find an appropriated ixed loader. The default action is just to print an error message "This script is protected by SourceGuardian™ and requires file ... " and stop executing. You may use any HTML text or PHP code here and it will be displayed or executed as a replacement to the default SourceGuardian™ loader error. This code WILL NOT BE ENCODED. This may be either HTML text or PHP code. For PHP code - you should use <?php ?> tags.

Do not append loader code

You may use this option if you don't want to include a default loader into the protected script. Any script encoded using this option will not be able to automatically find and load the appropriate ixed loader and you have to install the ixed loader manually to run this script. See the section about the manual ixed installation.

NOTE: if you select this option then Header options above have no effect (as they are placed inside this loader).

2.5.5 Binding



License mode
(●) Embedded — Project ID: 55a2b15f6b0149d768e1 — Project Key: 28102f42ca82701b50bd
( ) External — File name:

Expiry date
[✓] Scripts will timeout on: 13 Oct 2005 (in 3 days)
[ ] Use atomic clock servers
[✓] Lock the scripts to work only online

Binding
( ) Encrypt to IP
( ) Bind to IP(s)

IP	Mask
111.111.111.111	255.255.255.0

[✓] Bind to hostname

Hostname
myhost.com

[✓] Bind to MAC

MAC
11:22:33:44:55:66

Encoded scripts security
[ ] Work only with other files encoded with your copy of SourceGuardian

License mode:

Embedded - means that all binding information will be inside each encoded script. This is how SourceGuardian 2.x worked.

External - allows you to use the Script License Generator for specifying binding options for your project.

The Script License Generator is an external tool for creating script license files. A Script license file is required to run protected scripts encoded with this option.

Using the script license is the best way of encoding if you need to distribute one script or an entire project between different users, but need to use different restriction options for each user.

Scripts encoded with this option will require an external license file to run. Protected scripts will search for the license file in the current directory and all parent directories. So you may have one license file for an entire protected project located in the top project directory.

If the protected script cannot find the specified license file it will display the error message: "script requires ... file to run"

After encoding is finished you will be prompted to create a license for your encoded scripts if you set 'Automatically open External License generator' option in application settings.

File name

You need to specify license file name that your project will use. Specify exactly the same file name in license generator when you generate license for this project.

Project ID

This allows you to assign ID to your project to identify what license it should accept. Specify the same Project ID in license generator when you generate license for this project. This option is useful when you want to ship several products that uses external license so that each license would work only with its Project ID.

Project Key

This is used in pair with Project ID - required if you plan to use external license mode. Introduced in 5.0
New algorithm in 5.0 uses the idea of two keys. The first key (Project Id) is stored in the encrypted area of protected script and used to decrypt an external license file. The second key (Project Key) is stored in the license file and used to decrypt the bytecode from the protected script.
Using the new algorithm protects your product from creating a full working copy from the demo version by some people who may be interested in this. As to decrypt and run a protected script a true license file for the full version of your product is required. Otherwise it's impossible to decrypt and run a bytecode.
Project Id and Project Key values are required if external license protection method is selected.

Expiry date

'Scripts will timeout on' - click on calendar icon to choose the date you wish the script to expire. At the left side you will see the hint showing how many days are left from today's date.

The script will not run on and after the specified date and displays the error message: "script has expired".

Use atomic clock servers

If you use a time lock option for your scripts you may wish to let the script to get the world time from the online time service for expiry checks than using the server time. You may specify a list of time services in SourceGuardian settings.

Lock the scripts to work only online

If you do not use time lock option you still able to lock the scripts to work only online. Do to so, select this option. SourceGuardian will check if it is working online by accessing to atomic clock servers.

Binding

'Bind to IP(s)' - bind script to an ip/mask. The encoder will lock the script to run only from the specified IP address(es). The specified IP address mask will be applied to the real IP address before comparing. So you may use this option to lock the script to a multiple IP if mask is specified. If run from not allowed IP script will come with the error message: "script cannot run on this machine" You may add as many IP address/mask pairs as you want using editor. Press '+' button if you need to add another IP/Mask pair (or press down arrow on keyboard while in editor). Press '-' button if you want to delete current IP/Mask pair.

IP address mask 255.255.255.255 is used by default if not specified.

Encrypt to IP - bind and encrypt to ip/mask. The encoder will lock the script to run only from the specified IP address. The encoder will use a specified IP address with applied mask as a part of the key for encryption for the maximum protection. The Loader will not be able to even decrypt a script from the wrong ip address and will display the error message: "script checksum error". IP address mask 255.255.255.255 is used by default if not specified. If you choose this option then the Bind to IP(s) option become inactive (and vice versa).

Bind to hostname: - you can bind the script to a domain name. The Encoder will lock the script to run only from the specified domain and all sub domains. If an attempt is made to run the script on a non-authorised domain, the following error message will be displayed: "script cannot run on this machine". You may add as many domain names as you want.

Hint: use the name of the main domain in this option, not the name of any sub domain until you are sure you need to lock to a sub domain.

Example 1: mydomain.com

The script will run from mydomain.com, www.mydomain.com, myname.mydomain.com etc but will NOT run from otherdomain.com, www.otherdomain.com, otherdomain.net etc.

Example 2: www.mydomain.com

Script will run ONLY from www.mydomain.com. It will not run on the Main domain mydomain.com and all other subdomains like myname.mydomain.com as well as other domains like otherdomain.com, www.otherdomain.com, otherdomain.net etc.

Bind to MAC - You can bind a script to LAN hardware (MAC) addresses. This address is unique for each networking adapter and so it may be easily used to identify a machine. A MAC address is 6 bytes long, with each byte represented in hex and separated with ':' or '-'.The encoder will lock a script to run only from the machine which has a networking adapter with the specified MAC address. If there is more than one LAN adapter installed then script will check all of them. If an attempt is made to run a script from a machine without the correct adapter, then the script will display the error message: "script cannot run on this machine" You may use this option more than once to specify multiple MAC addresses.

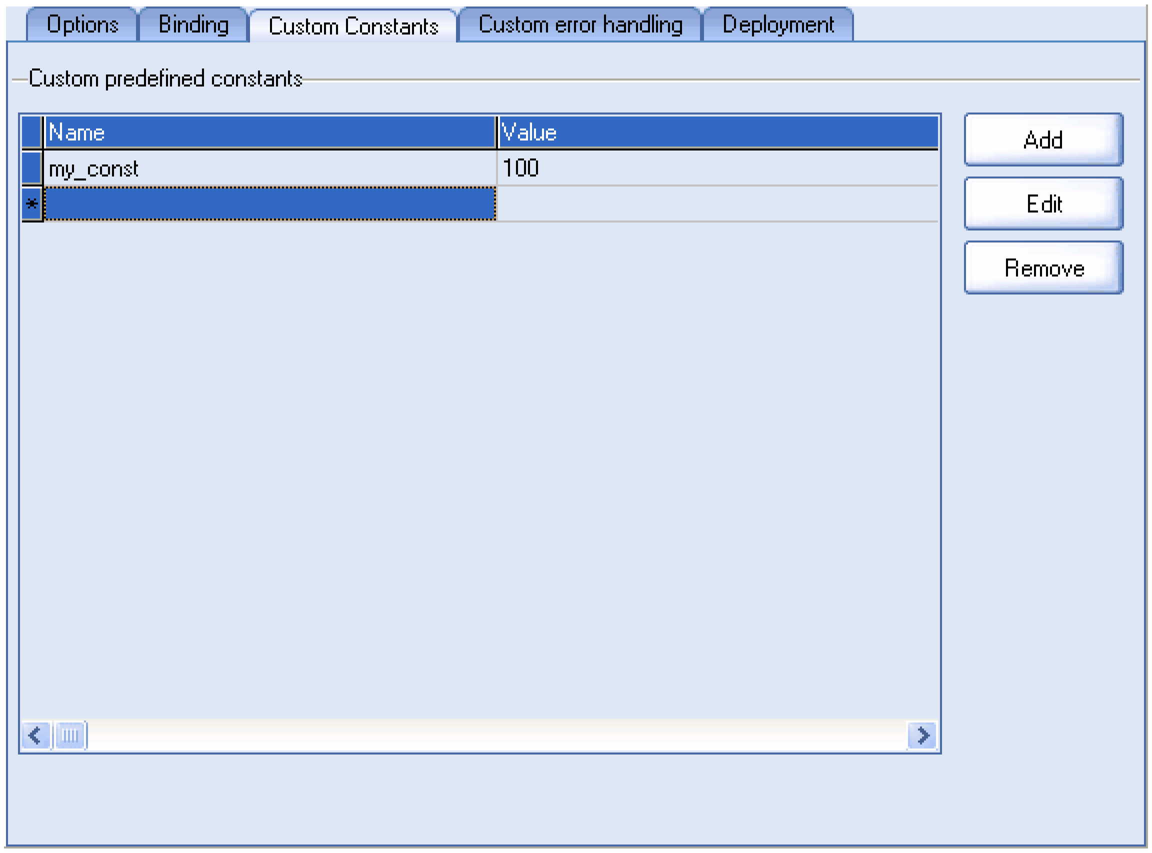
Hint: you may use 'ifconfig' command under Linux or 'route print' under Windows to get a list of installed networking adapters and known MAC addresses.

Encoded scripts security

Work only with other files encoded with your copy of SourceGuardian™ - script will work only with other encoded files. This option makes sense only when encoding multiple files. All scripts encoded with this option will work only with other encoded files and will NOT work if any of the included files or top files are substituted with an unencoded one or encoded by another installation of SourceGuardian™ 5.0 for PHP. This gives you the ultimate protection for your projects when multiple PHP scripts are used together.

Example: If you have a password in a.php and then b.php includes a.php and calls c.php for any action. No one can substitute c.php with their own code and do 'echo $password' to know your password if this option was used during the encoding of all files. Also no one can create d.php which will include protected a.php and then do 'echo $password'.

2.5.6 Custom Constants



Name	Value
my_const	100

SourceGuardian let you define custom named constants during encoding process or within an external script license. Constant name/value pairs are stored internally in the encrypted area of the protected script or external
license. They may be used for custom script locking or any other actions if you need to store a custom value in protected script or script license file and then retrieve it from your PHP code.

To get a predefined constant value from the PHP code use sg_get_const() function. This function is defined in SourceGuardian loader.

Syntax: string sg_get_const(string)

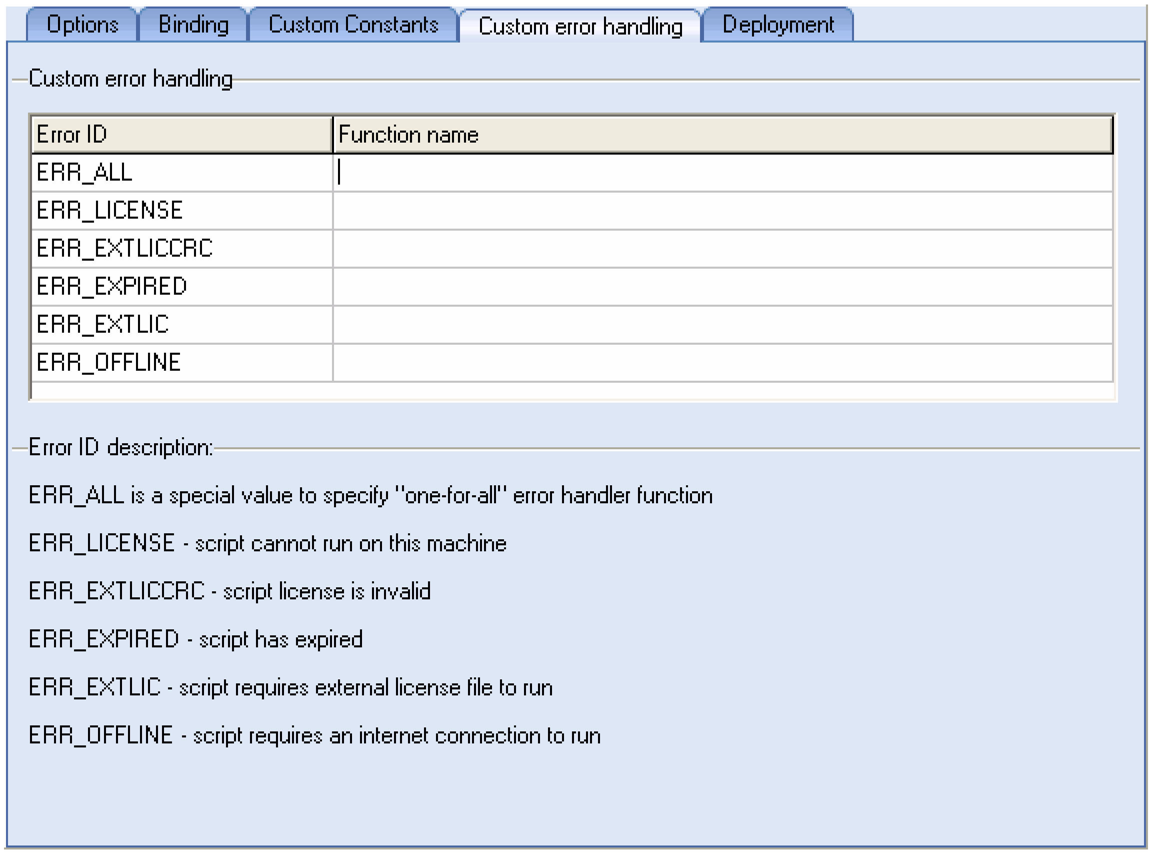
> Will return a predefined SourceGuardian constant value or FALSE if constant with the specified name is not defined. SourceGuardian constants names are *case sensitive*.

There are 5 constants predefined for each protected script:

sg_get_const("encoder")	returns the name of encoder "SourceGuardian"
sg_get_const("version")	returns the encoder version number
sg_get_const("encode_date")	return UNIX timestamp when the script was encoded
sg_get_const("license_date")	return UNIX timestamp when the script license was created. It's may differ from "encode_date" when external script license is used

sg_get_const("expire_date")	returns script expiration date as UNIX timestamp if it's defined in the script license or internally via script binding options during encoding

2.5.7 Custom error handling



Custom error handling

Error ID	Function name
ERR_ALL	
ERR_LICENSE	
ERR_EXTLICCRC	
ERR_EXPIRED	
ERR_EXTLIC	
ERR_OFFLINE	

Error ID description:

ERR_ALL is a special value to specify "one-for-all" error handler function

ERR_LICENSE - script cannot run on this machine

ERR_EXTLICCRC - script license is invalid

ERR_EXPIRED - script has expired

ERR_EXTLIC - script requires external license file to run

ERR_OFFLINE - script requires an internet connection to run

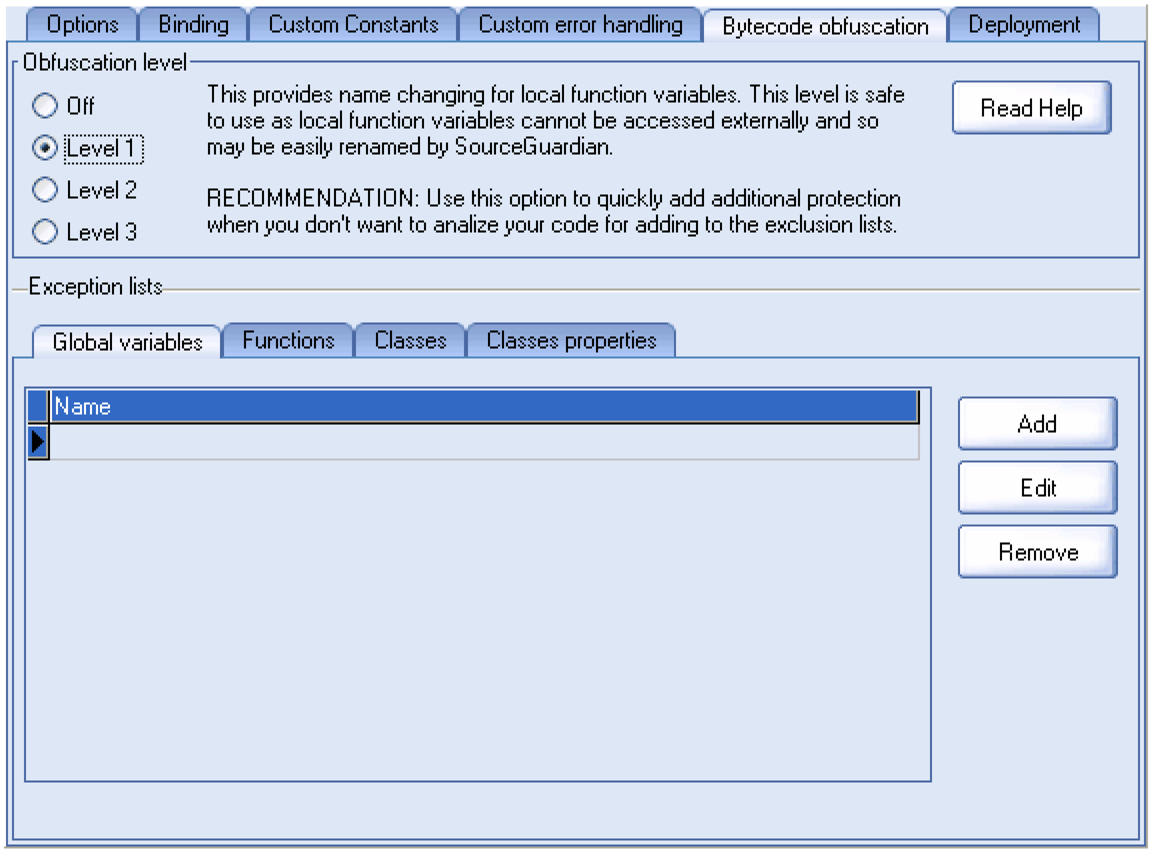
You may add custom error handling functions which will catch script licensing errors. Error handler should be a function which accepts two parameters:

sg_error_handler(code , message)

You may use any name for this function. Also you may have different functions for different script errors. The first argument will contain a error code. The second one will contain a default error message.

Custom error handler function should be defined before an error may occur. The best place for it is in "prepend header" code as it's loaded *before* any license checking is done and so error handler will be always available if defined here. But you may also define a custom error handler function in another encoded file which will be included before the script which may cause a license error. Don't put any passwords etc secret data if you use a "prepend header" code for defining a error handler as this code is stored unencoded.

2.5.8 Bytecode obfuscation



Obfuscation level

( ) Off
[✓ Level 1]
( ) Level 2
( ) Level 3

This provides name changing for local function variables. This level is safe to use as local function variables cannot be accessed externally and so may be easily renamed by SourceGuardian.

RECOMMENDATION: Use this option to quickly add additional protection when you don't want to analize your code for adding to the exclusion lists.

Additional bytecode protection

Since version 6.0 of SourceGuardian has new options for stronger bytecode protection. In addition to our standard bytecode compiling, encrypting and compressing, the new encoder can obfuscate the names of variables, functions and classes within
the bytecode to make it an unreadable value.

Additional bytecode protection may be set from levels 1 to 3. For levels 2 and 3 it will require a greater knowledge of your scripts and will require some manual intervention to exclude some variables, functions or classes from names changing - this may be necessary to ensure your application works as it should do.

If no bytecode protection level has been selected, then the encoder will work without obfuscation, but your scripts will still be compiled into bytecode, encrypted and compressed.

Levels of bytecode protection

Each level of bytecode protection includes the functionality of the previous one, but also adds more protection.

Please see the section "Bytecode Obfuscation". By default we have chosen "Bytecode with Obfuscation Level 1" but you can change it to any setting you wish.

Level 1 obfuscation - This provides name changing for local function variables. This level is safe to use as local function variables cannot be accessed externally and so may be easily renamed by SourceGuardian.

RECOMMENDATION: Use this option to quickly add additional protection when you don't want to analyze your code for adding to the exclusion lists.

Level 2 obfuscation - This level additionally changes the names of all global vars, function names and class names defined within the file.

RECOMMENDATION:

1) if register_globals is used and EGPCS vars are used directly then their names should not be changed. You need to put such global variables names into the global variables exclusion list (see below)

2) global variables accessed via the $GLOBALS array by name should keep their names and so need to be put into global vars exclusion list. By default names will be changed for all global variables and they may still be accessed correctly if within the project even if they are in different files.

3) global variables used in the project and accessed externally from other unencoded (or encoded with other options) scripts should keep their names and they should not be obfuscated so they need to be put into the global vars exclusion list.

4) possible errors could occur if a function or a class is defined in any script that is used in another script. If so put this function name or class name into the appropriate exclusion list.

Example for this level only:

```
a.php: <?php function myfunc() { echo 'test'; } ?>
b.php: <?php include "a.php"; myfunc(); ?>
```

The myfunc() function above is defined in one file and is also used in another one. To run under level 2 you need to put the "myfunc" function name in to the functions exclusion list.

Use this option for encoding separate scripts or projects which have not much code interaction between separate scripts. Each function or class defined in one script and used in another script should not change its name and so needs to be put in the exclusion list.

Level 3 obfuscation - additionally this changes the names of all functions, classes, class methods and class properties.

RECOMMENDATION:

Same as for level 2:

1) if register_globals is used and EGPCS vars is used directly then their names should not be changed. You need to put such global variables names into the global variables exclude list (see below)

2) global variables are accessed via the $GLOBALS array by name should keep their names and so to be put into global vars exclusion list.

3) global variables used in the project and accessed externally from other unencoded (or encoded with other options) scripts should keep their names so need to be put into the global vars exclusion list. By default names will be changed for all global variables and they still may be accessed OK if

within the project even in different files.

New to level 3:

4) names of functions and classes defined in the extensions should not be changed and so need to be defined in the exclusion lists. ex.
   mysql_connect(), mysql_select_db(), mysql_query()...(mysql),
   preg_match()...(pcre), MimeMessage class (mailparse) etc.

This option will change the names for the user defined or extended functions and classes. It will not change the names for standard built-in functions such as print(), str_replace(), etc and built-in classes such as Exception, Iterator etc.

One other difference from level 2 is that as names will be changed for all functions and classes by default then there is no problem for the script to include a file from the same project and call a function defined in the included file or to create an object of the class defined there.

Example for this level only:

```
a.php: <?php function myfunc($arg) { return preg_match('/test/',$arg); } ?>
b.php: <?php include "a.php"; myfunc(); ?>
```

preg_match() is not a standard PHP function. This function is defined in separate pcre extension and so its name "preg_match" should be put into the functions exclusion list to run under level 3. Please note, there is no problem with the
myfunc() function being defined in one file and used in another one under level 3.

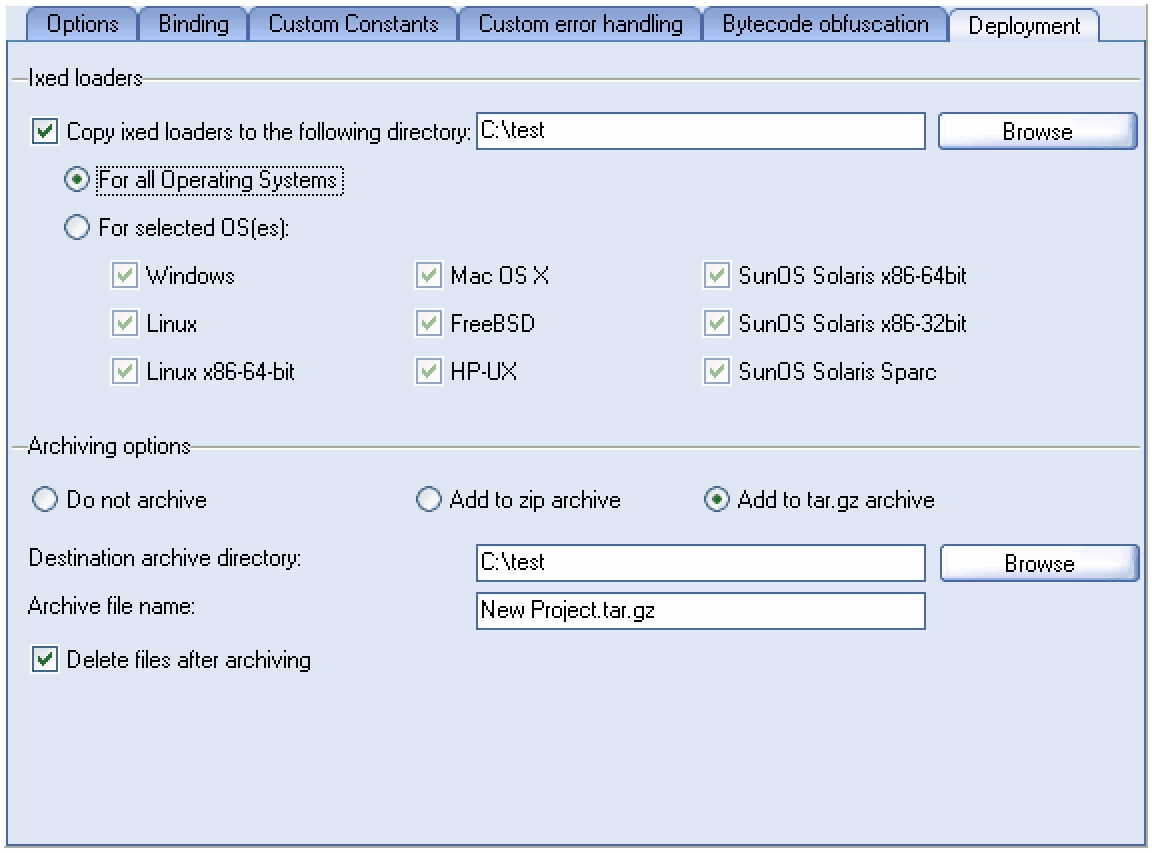
Use this option for encoding the whole projects which will include everything needed to run your application and which has no unencoded functions or classes. There are no problems to still have a custom settings file as such settings are usually done via assigning global vars, constants or even ini files.

RECOMMENDATION for all levels of bytecode protection; Except for level 1 the encoder does significant changes to the bytecode this creates additional protection. You should test your encoded projects well and understand the risk of this kind of additional protection that if not used correctly you may have obfuscated and unobfuscated information that will then not work well together. Analysis is definitely required for using levels 2 and 3 and this cannot be done automatically by the encoder. Therefore you need to fill in the exclusion lists for global variables, functions, classes and class properties according to your code.

Bytecode protection exclusion lists

For bytecode protection levels 2 and 3 you need to specify the names of global variables, functions, classes as well as class properties for which names should not be changed. Use Exception lists for this. As usual you can Add/Edit/Remove items.

2.5.9 Deployment



Ixed loaders

[✓] Copy ixed loaders to the following directory: C:\test

(●) For all Operating Systems

( ) For selected OS(es):

[✓] Windows [✓] Mac OS X [✓] SunOS Solaris x86-64bit

[✓] Linux [✓] FreeBSD [✓] SunOS Solaris x86-32bit

[✓] Linux x86-64-bit [✓] HP-UX [✓] SunOS Solaris Sparc

Archiving options

( ) Do not archive ( ) Add to zip archive (●) Add to tar.gz archive

Destination archive directory: C:\test

Archive file name: New Project.tar.gz

[✓] Delete files after archiving

Ixed loaders

'Copy ixed loaders' - allows you to specify a directory where you want to copy the ixed loaders after the encoding process. This field is filled automatically when you change the 'Encode to a target directory' field value. You can change it to any other path if you wish. If the directory doesn't already exist then it will be automatically created.

You can choose to copy ixed loaders for all Operating Systems or select what systems you need.

NOTE: loaders will be copied only for target PHP version that you selected for this project (4.x or 5.x).

Archiving options

This section allows you to select archiving format (zip or tar.gz). Default option is 'Do not archive'. Destination archive directory is the same as target encoding directory. Its recommended to leave itunchangedd if you plan to upload archived project via GUI installer to your local or remoteweb serverr.
'Delete files after archiving' - after archive is created all files are removed from target directory (files are moved to archive).

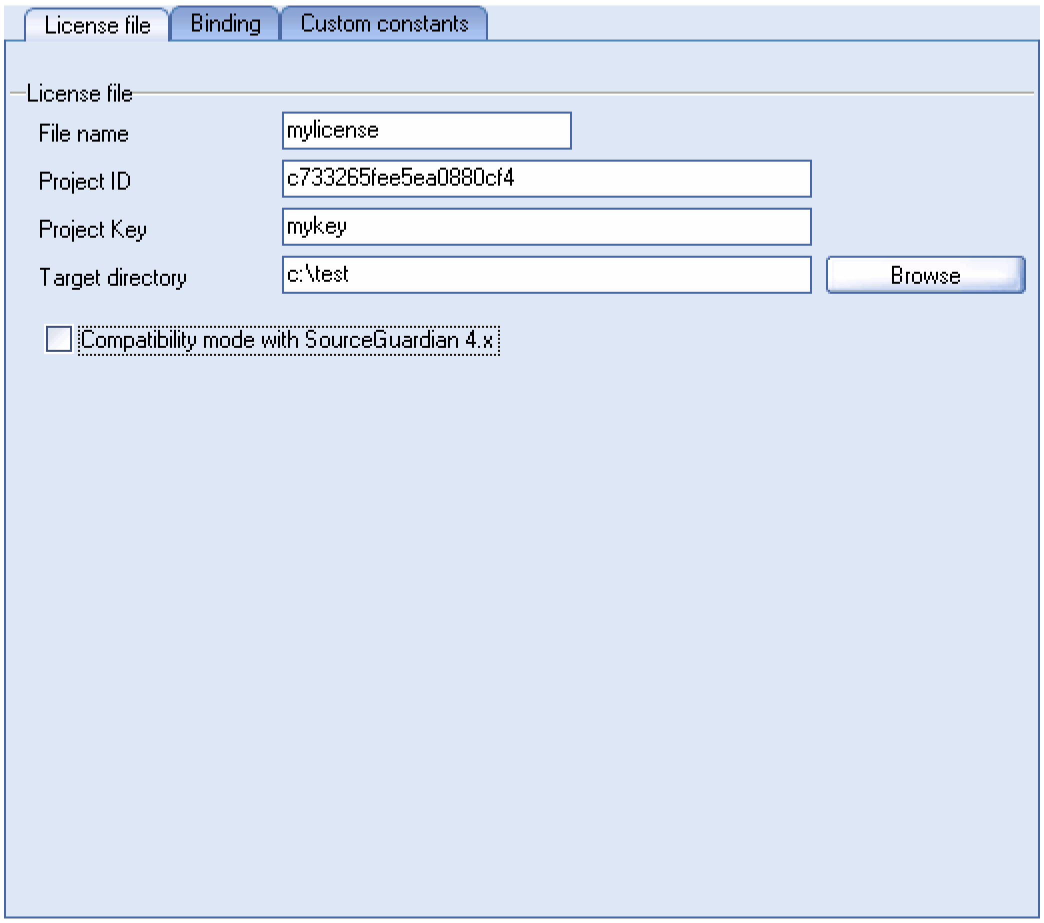
2.5.10 License generator

The Script License Generator is an external tool for creating script license files. A Script license file is required to run protected scripts encoded with this option.

Using the script license is the best way of encoding if you need to distribute one script or entire project between different users but need to use different restriction options for each user.

Scripts encoded with this option will require an external license file to run. Protected scripts will search for the license file in the current directory and all parent directories. So you may have one license file for an entire protected project located in the top project directory.

If the protected script cannot find the specified license file it will display the error message: "script requires ... file to run"



License file

File name: mylicense

Project ID: c733265fee5ea0880cf4

Project Key: mykey

Target directory: c:\test

☐ Compatibility mode with SourceGuardian 4.x

License file

File name - the name of license file. Should be the same as you specified during encoding of your scripts!

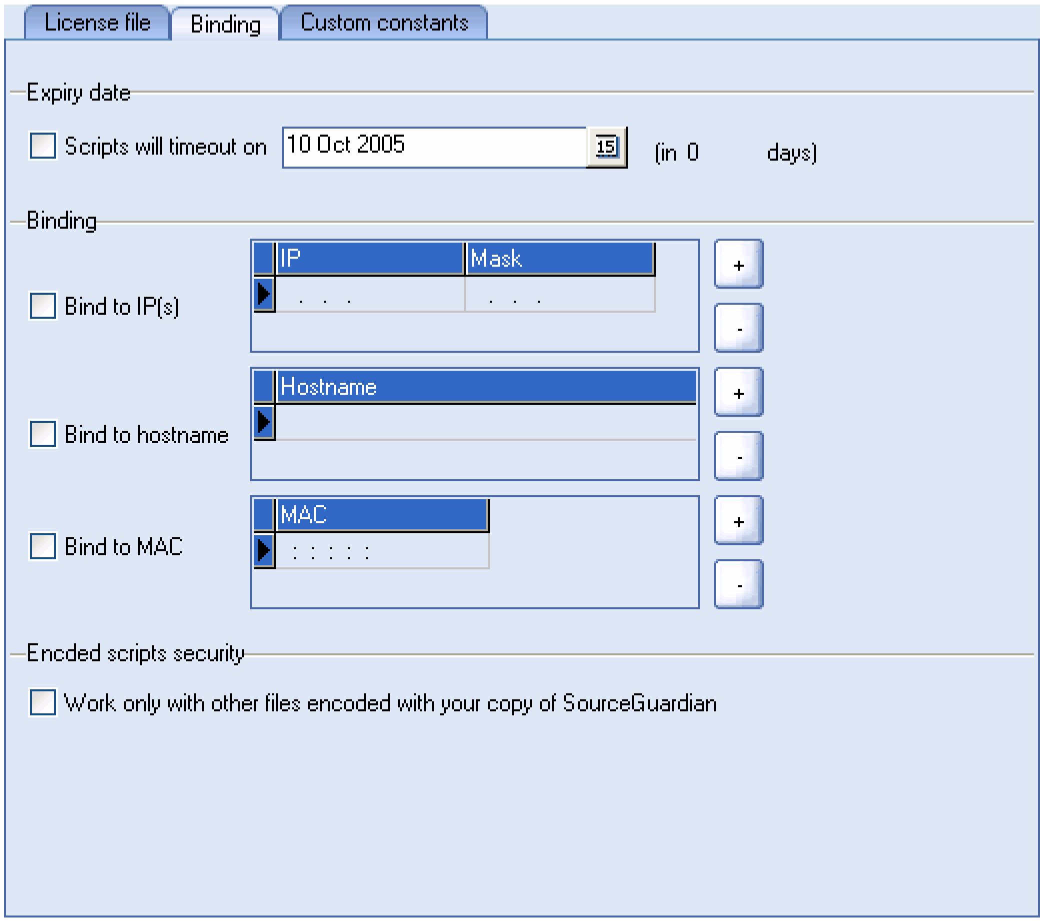
Project ID - this allows you to assign ID to your project to identify what license it should accept. Specify the same Project ID in license generator when you generate license for this project. This option is useful when you want to ship several products that uses external license so that each license would work only with its Project ID.

Project Key - this is used in pair with Project ID - required if you plan to use external license mode. New algorithm in 5.0 uses the idea of two keys. The first key (Project Id) is stored in the encrypted area of protected script and used to decrypt an external license file. The second key (Project Key) is stored in the license file and used to decrypt the bytecode from the protected script.
Using the new algorithm protects your product from creating a full working copy from the demo version by some people who may be interested in this. As to decrypt and run a protected script a true

license file for the full version of your product is required. Otherwise it's impossible to decrypt and run a bytecode.
Project Id and Project Key values are required if external license protection method is selected.

Target directory - directory path where you put the generated license file.

Compatibility mode with SourceGuardian 4.x - select this if you want to generate license for scripts encoded with SourceGuardian 4.x



Expiry date

- [ ] Scripts will timeout on 10 Oct 2005 (in 0 days)

Binding

- [ ] Bind to IP(s)

IP	Mask
. . .	. . .

- [ ] Bind to hostname

Hostname

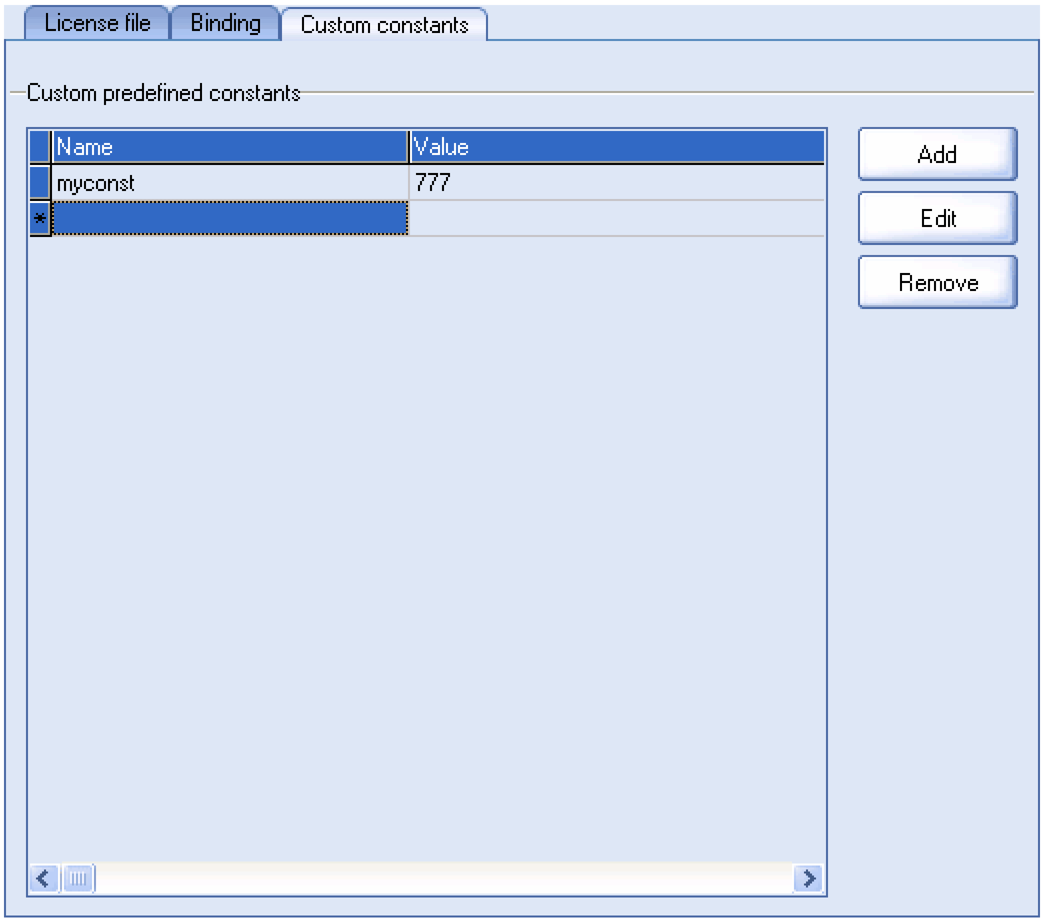
- [ ] Bind to MAC

MAC
: : : : :

Encded scripts security

- [ ] Work only with other files encoded with your copy of SourceGuardian

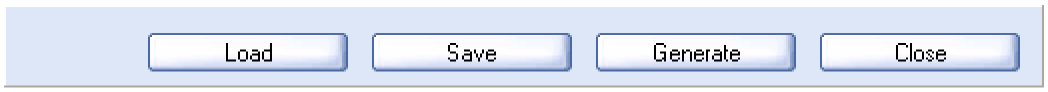
Binding - refer to Binding Options chapter for more detailed information about them.



Name	Value
myconst	777

Custom constants - refer to Custom constants chapter for more detailed information about them.

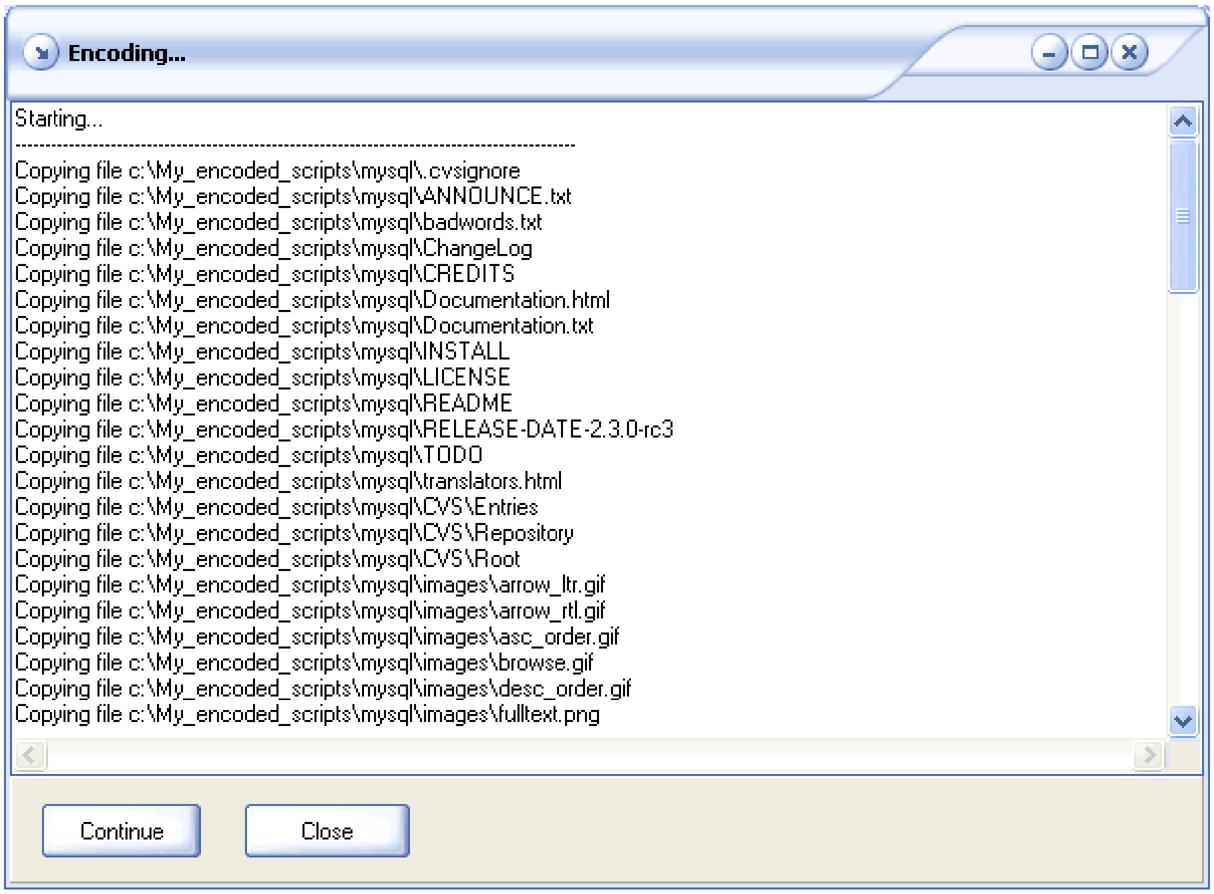
At the bottom of licgen window you can see these buttons:



'Save' - allows you to save license data so that you can open it next time using 'Load' button.
'Generate' - generates license file.
'Close' - closes license generator window.

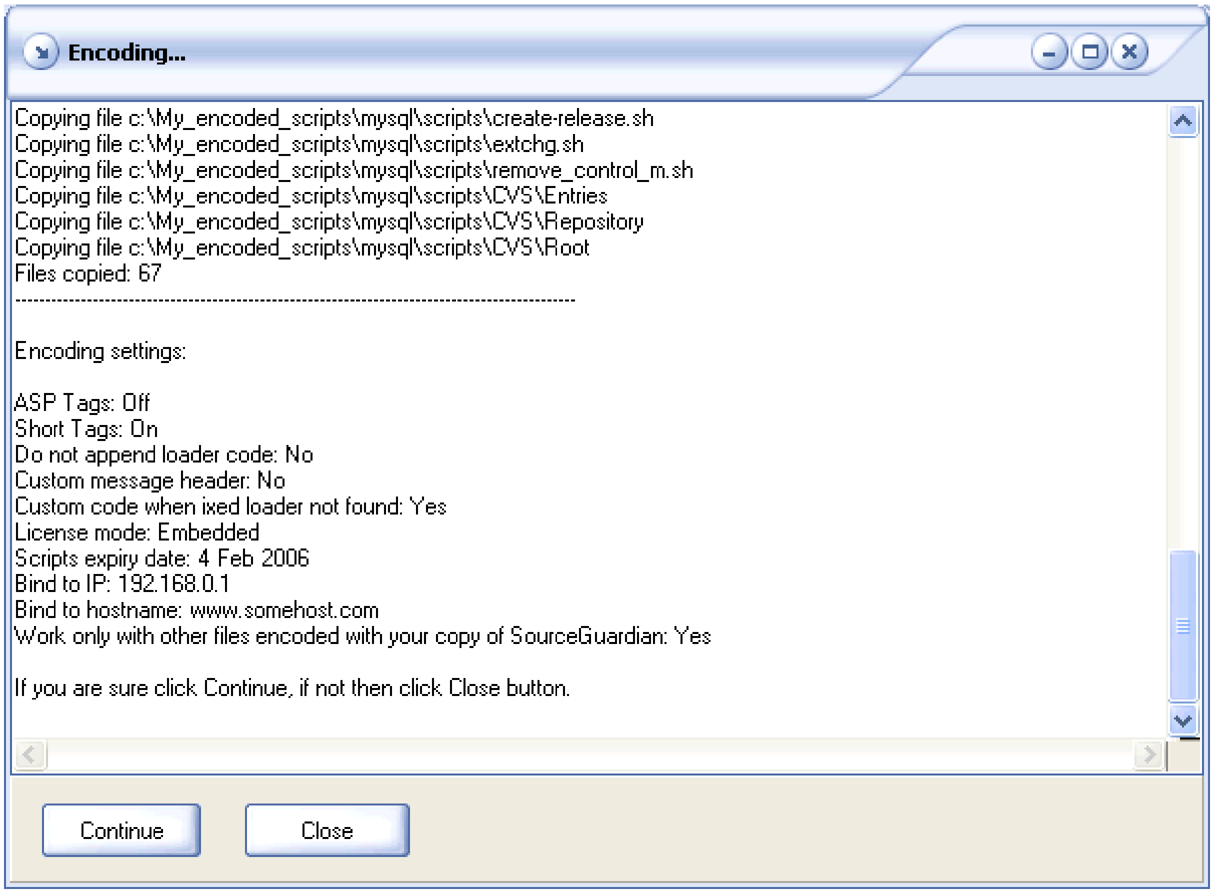
2.6 Encoding process

At first step SourceGuardian™ copies all files (if any) to a target directory:



```
Starting...
------------------------------------------------------------------------------------------------
Copying file c:\My_encoded_scripts\mysql\.cvsignore
Copying file c:\My_encoded_scripts\mysql\ANNOUNCE.txt
Copying file c:\My_encoded_scripts\mysql\badwords.txt
Copying file c:\My_encoded_scripts\mysql\ChangeLog
Copying file c:\My_encoded_scripts\mysql\CREDITS
Copying file c:\My_encoded_scripts\mysql\Documentation.html
Copying file c:\My_encoded_scripts\mysql\Documentation.txt
Copying file c:\My_encoded_scripts\mysql\INSTALL
Copying file c:\My_encoded_scripts\mysql\LICENSE
Copying file c:\My_encoded_scripts\mysql\README
Copying file c:\My_encoded_scripts\mysql\RELEASE-DATE-2.3.0-rc3
Copying file c:\My_encoded_scripts\mysql\TODO
Copying file c:\My_encoded_scripts\mysql\translators.html
Copying file c:\My_encoded_scripts\mysql\CVS\Entries
Copying file c:\My_encoded_scripts\mysql\CVS\Repository
Copying file c:\My_encoded_scripts\mysql\CVS\Root
Copying file c:\My_encoded_scripts\mysql\images\arrow_ltr.gif
Copying file c:\My_encoded_scripts\mysql\images\arrow_rtl.gif
Copying file c:\My_encoded_scripts\mysql\images\asc_order.gif
Copying file c:\My_encoded_scripts\mysql\images\browse.gif
Copying file c:\My_encoded_scripts\mysql\images\desc_order.gif
Copying file c:\My_encoded_scripts\mysql\images\fulltext.png
```

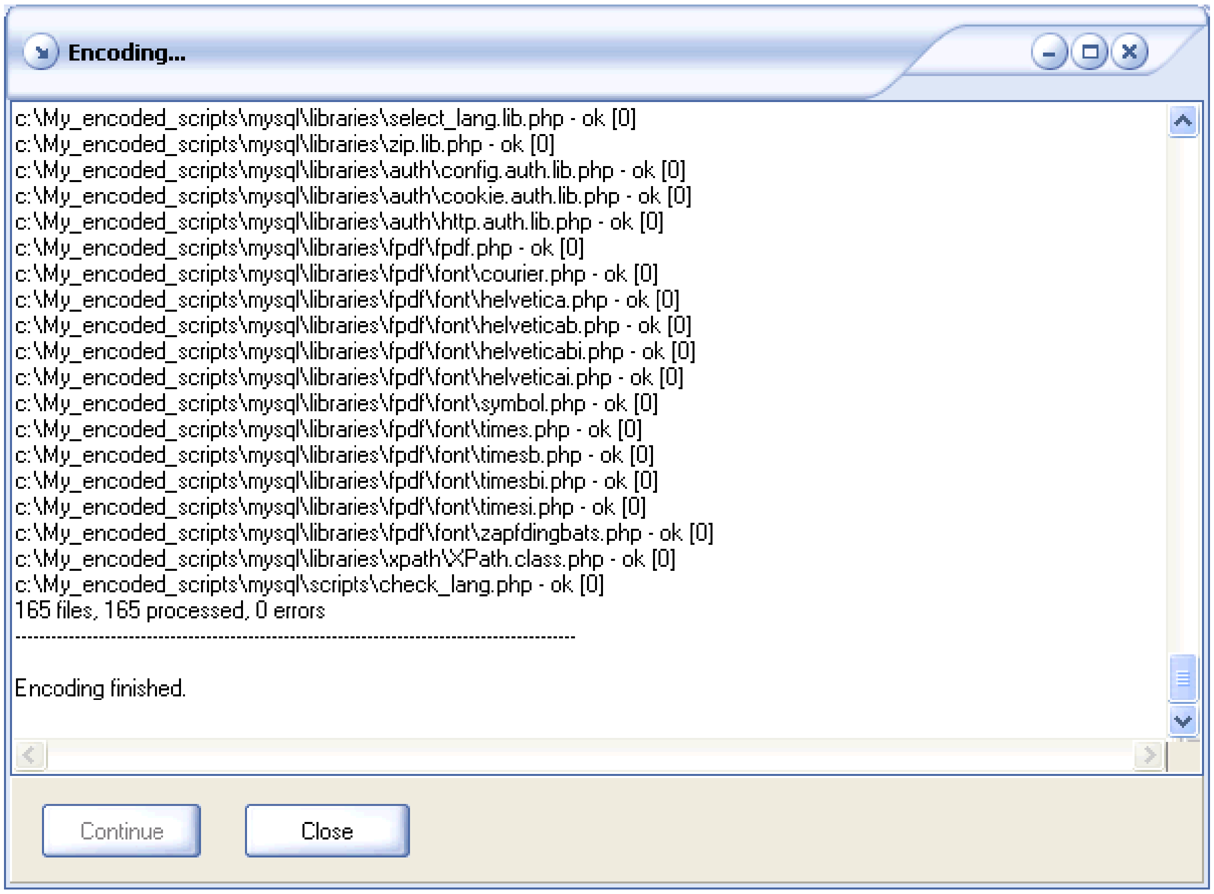
Then it prints Encoding settings so that you can check them once again and if you think that all is Ok click on 'Continue' button. If not click on 'Close'.



```
Copying file c:\My_encoded_scripts\mysql\scripts\create-release.sh
Copying file c:\My_encoded_scripts\mysql\scripts\extchg.sh
Copying file c:\My_encoded_scripts\mysql\scripts\remove_control_m.sh
Copying file c:\My_encoded_scripts\mysql\scripts\CVS\Entries
Copying file c:\My_encoded_scripts\mysql\scripts\CVS\Repository
Copying file c:\My_encoded_scripts\mysql\scripts\CVS\Root
Files copied: 67
-----------------------------------------------------------------------------------------

Encoding settings:

ASP Tags: Off
Short Tags: On
Do not append loader code: No
Custom message header: No
Custom code when ixed loader not found: Yes
License mode: Embedded
Scripts expiry date: 4 Feb 2006
Bind to IP: 192.168.0.1
Bind to hostname: www.somehost.com
Work only with other files encoded with your copy of SourceGuardian: Yes

If you are sure click Continue, if not then click Close button.
```

When you click on 'Continue' encoding starts. The mouse cursor will change its state to indicate an encoding process. Please wait for encoding to complete.

At the end of the encoding process you will see a listing of files with a status of encoding for each file and a summary at the end.



```
c:\My_encoded_scripts\mysql\libraries\select_lang.lib.php - ok [0]
c:\My_encoded_scripts\mysql\libraries\zip.lib.php - ok [0]
c:\My_encoded_scripts\mysql\libraries\auth\config.auth.lib.php - ok [0]
c:\My_encoded_scripts\mysql\libraries\auth\cookie.auth.lib.php - ok [0]
c:\My_encoded_scripts\mysql\libraries\auth\http.auth.lib.php - ok [0]
c:\My_encoded_scripts\mysql\libraries\fpdf\fpdf.php - ok [0]
c:\My_encoded_scripts\mysql\libraries\fpdf\font\courier.php - ok [0]
c:\My_encoded_scripts\mysql\libraries\fpdf\font\helvetica.php - ok [0]
c:\My_encoded_scripts\mysql\libraries\fpdf\font\helveticab.php - ok [0]
c:\My_encoded_scripts\mysql\libraries\fpdf\font\helveticabi.php - ok [0]
c:\My_encoded_scripts\mysql\libraries\fpdf\font\helveticai.php - ok [0]
c:\My_encoded_scripts\mysql\libraries\fpdf\font\symbol.php - ok [0]
c:\My_encoded_scripts\mysql\libraries\fpdf\font\times.php - ok [0]
c:\My_encoded_scripts\mysql\libraries\fpdf\font\timesb.php - ok [0]
c:\My_encoded_scripts\mysql\libraries\fpdf\font\timesbi.php - ok [0]
c:\My_encoded_scripts\mysql\libraries\fpdf\font\timesi.php - ok [0]
c:\My_encoded_scripts\mysql\libraries\fpdf\font\zapfdingbats.php - ok [0]
c:\My_encoded_scripts\mysql\libraries\xpath\XPath.class.php - ok [0]
c:\My_encoded_scripts\mysql\scripts\check_lang.php - ok [0]
165 files, 165 processed, 0 errors
--------------------------------------------------------------------------------------------

Encoding finished.
```

Since 7.0 you will see 'Encoding finished' popup after encoding is finished.

Click on 'Close' button.

NOTE: If you have chosen the External license option, then the License Generator dialogue will appear after you press the 'Close' button.

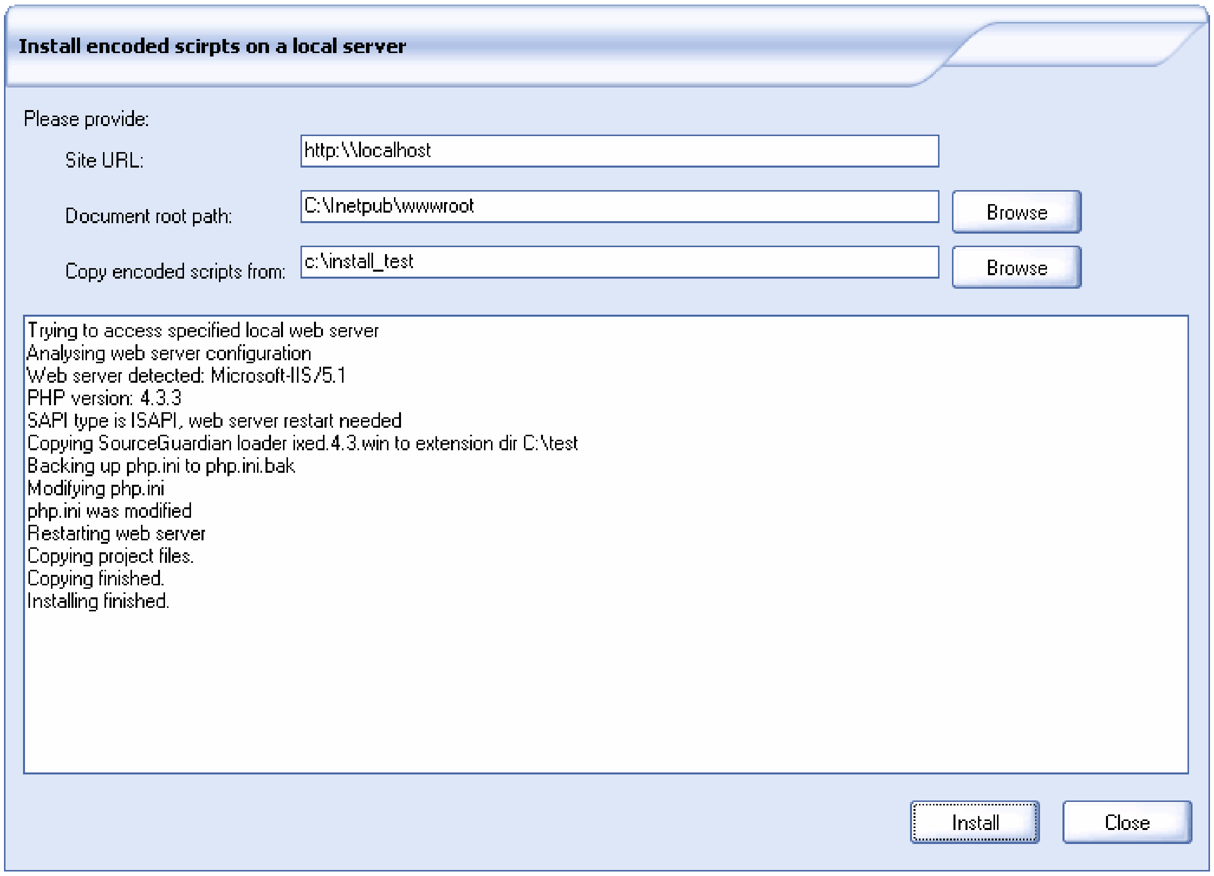
NOTE: Unless you disabled it, encoded scripts installer on local server dialogue will appear after you press the 'Close' button.

NOTE: Unless you disabled it, encoded scripts installer on remote server dialogue will appear after you press the 'Close' button.

2.7 Installers

There are two separate install wizards that can be used to install encoded scripts/loaders to a local web server or remote server.

Install encoded scripts on a local web server



Site URL: http:\\localhost

Document root path: C:\Inetpub\wwwroot

Copy encoded scripts from: c:\install_test

```
Trying to access specified local web server
Analysing web server configuration
Web server detected: Microsoft-IIS/5.1
PHP version: 4.3.3
SAPI type is ISAPI, web server restart needed
Copying SourceGuardian loader ixed.4.3.win to extension dir C:\test
Backing up php.ini to php.ini.bak
Modifying php.ini
php.ini was modified
Restarting web server
Copying project files.
Copying finished.
Installing finished.
```

Site URL - URL to your site

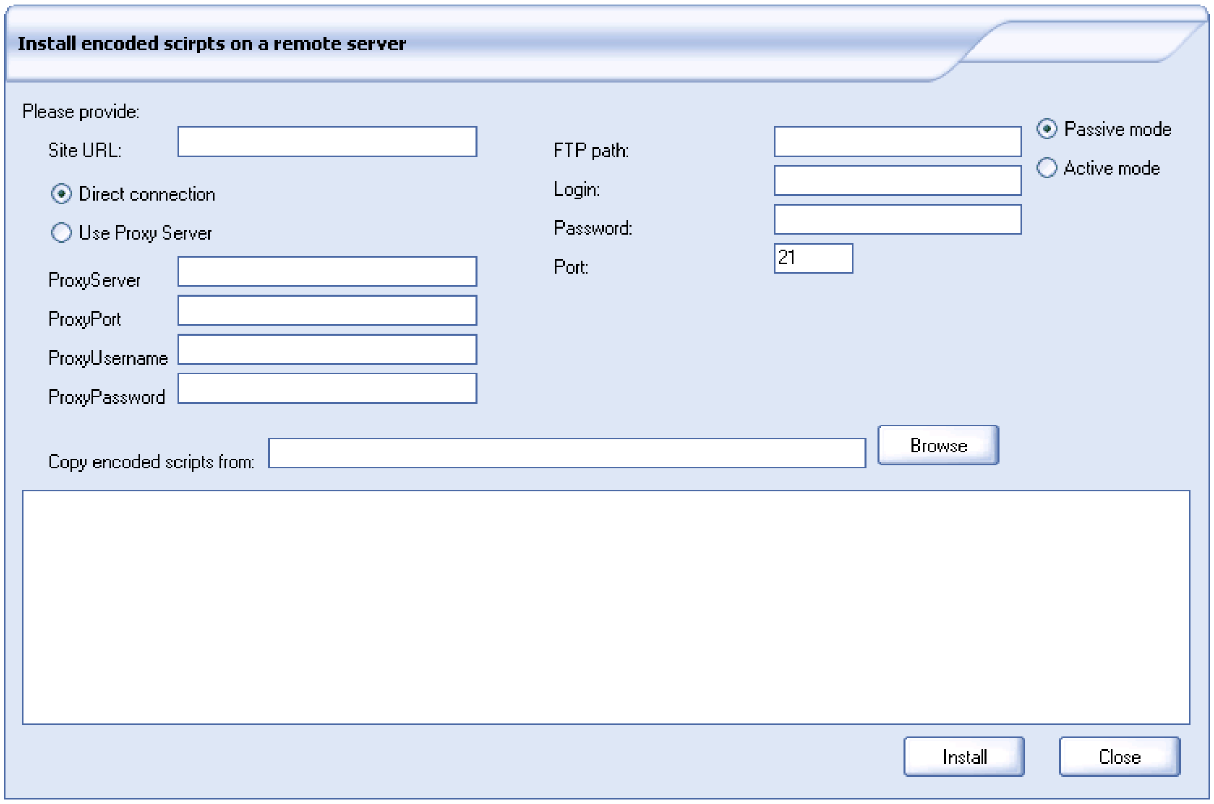
Document root path - full path to document root of your site

Copy encoded scripts from - path from which installer will copy all of its contents. Filled automatically if you come to this installer after encoding. Leave blank if you do not want to copy any encoded files and then only SourceGuardian loader will be installed.

Click on 'Install' button and you will see progress in log window.

Install wizard can automatically restart Apache and Microsoft-IIS web servers. If you have some different web server then you may need to restart it manually. If you are still unsure how to do it then just reboot your PC.

Install encoded scripts on a remote web server


Install encoded scirpts on a remote server

Please provide:

Site URL:

[✓ Direct connection] Use Proxy Server

ProxyServer

ProxyPort

ProxyUsername

ProxyPassword

FTP path:

Login:

Password:

Port: 21

[✓ Passive mode] Active mode

Copy encoded scripts from:

Site URL - URL to your site

Direct connection/Use Proxy Server - choose this according to internet connection that you have. Fill Proxy server details below if needed.

FTP Path - ftp server address and path where you want to install encoded scripts. For example: mysite.com\public_html

Login - login for this ftp

Password - password for this ftp

Port - ftp server port (default is 21)

Passive/Active mode - ftp mode to access ftp server

Copy encoded scripts from - path from which installer will copy all of its contents. Filled automatically if you come to this installer after encoding. Leave blank if you do not want to copy any encoded files and then only SourceGuardian loader will be installed.

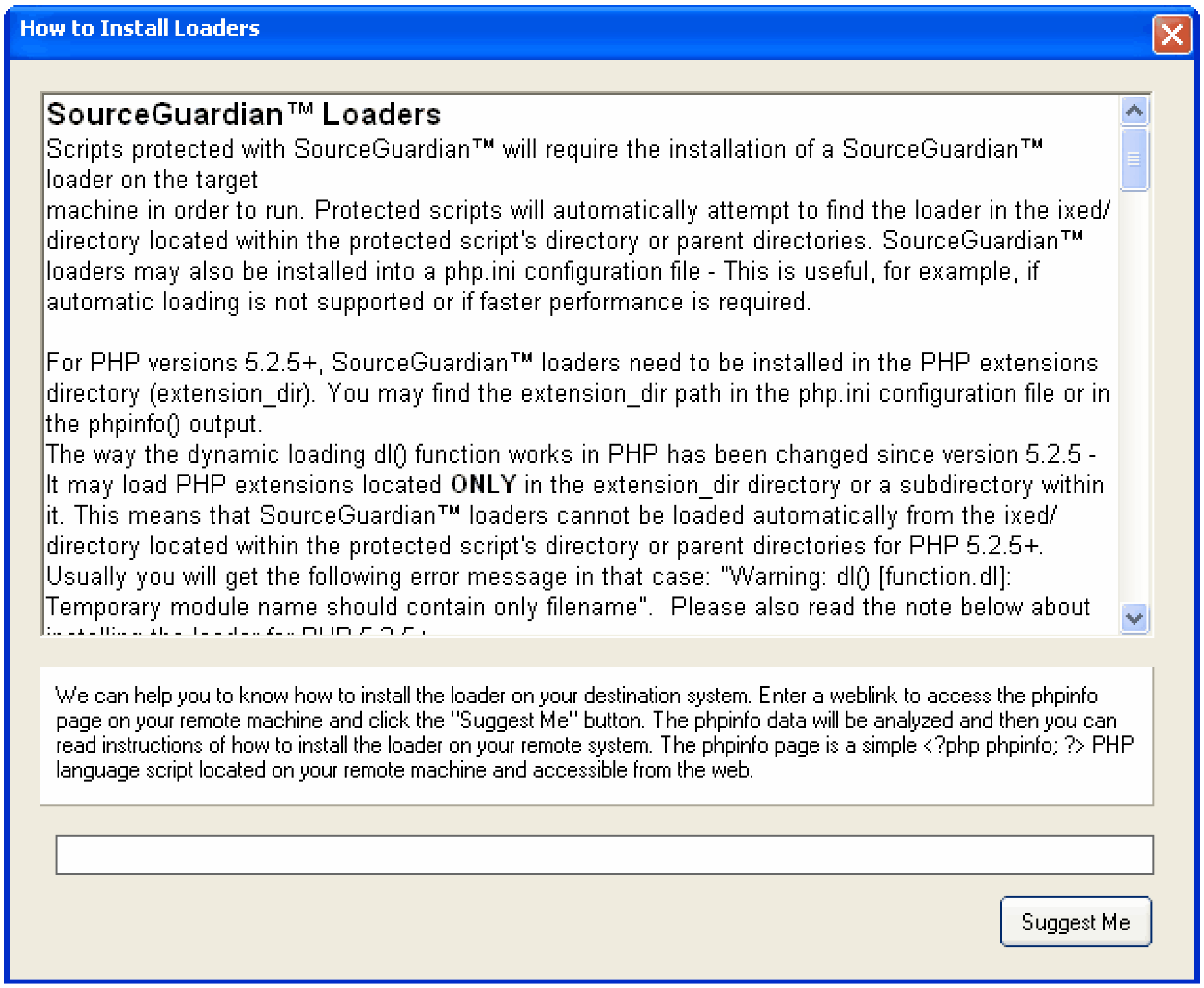
Click on 'Install' button and you will see progress in log window. If for example there are two types loader that suit to your web server (for example 32-bit and 64-bit loaders) then you will be asked which one you would like to install.

2.8 How to Install Loaders

How to install the loader helper

We can help you to know how to install the loader on your destination system. You need to create a simple phpinfo script and install it on your server. The content is simple as "<?php phpinfo(); ?>".

Then **enter below** a weblink to access the phpinfo page on your remote machine and click the "Suggest Me" button. The phpinfo data will be analyzed and then you can read instructions of how to install the loader on your remote system.



SourceGuardian™ Loaders

Scripts protected with SourceGuardian™ will require the installation of a SourceGuardian™ loader on the target
machine in order to run. Protected scripts will automatically attempt to find the loader in the ixed/ directory located within the protected script's directory or parent directories. SourceGuardian™ loaders may also be installed into a php.ini configuration file - This is useful, for example, if automatic loading is not supported or if faster performance is required.

For PHP versions 5.2.5+, SourceGuardian™ loaders need to be installed in the PHP extensions directory (extension_dir). You may find the extension_dir path in the php.ini configuration file or in the phpinfo() output.
The way the dynamic loading dl() function works in PHP has been changed since version 5.2.5 - It may load PHP extensions located **ONLY** in the extension_dir directory or a subdirectory within it. This means that SourceGuardian™ loaders cannot be loaded automatically from the ixed/ directory located within the protected script's directory or parent directories for PHP 5.2.5+. Usually you will get the following error message in that case: "Warning: dl() [function.dl]: Temporary module name should contain only filename". Please also read the note below about

We can help you to know how to install the loader on your destination system. Enter a weblink to access the phpinfo page on your remote machine and click the "Suggest Me" button. The phpinfo data will be analyzed and then you can read instructions of how to install the loader on your remote system. The phpinfo page is a simple <?php phpinfo; ?> PHP language script located on your remote machine and accessible from the web.

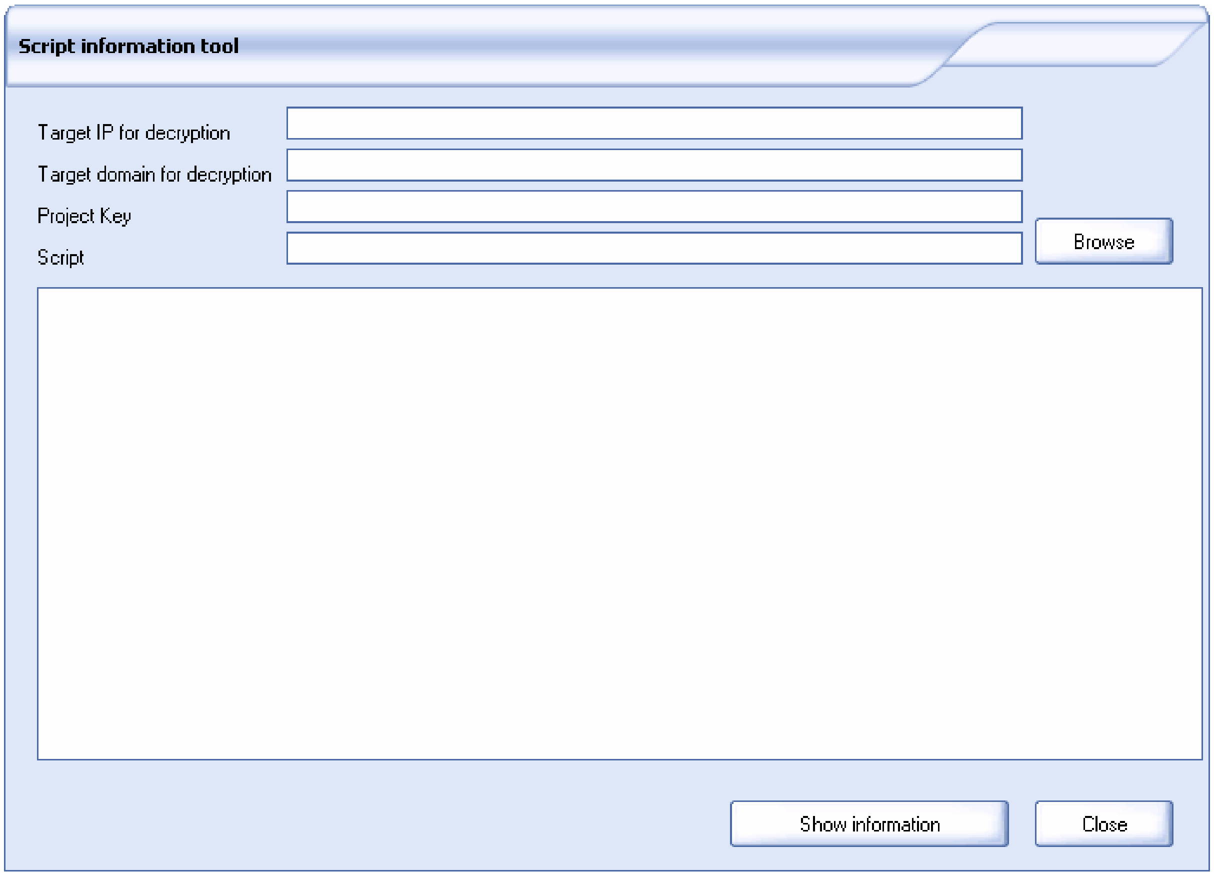
2.9 Information tools

You may get information about protected script or external script license. This may be useful for supporting your customers, checking scripts or licenses passed to them etc. You may know the date of encode, expiration date, binding options etc parameters from the protected script or script license.

It's possible to display script information only for scripts created *with the same installation* of SourceGuardian. To display script license information from an external file you need to know and specify the project id.

In GUI two separate dialogues available - for information about scripts and licenses.

Script information tool



Target IP for decryption - required if script was encoded using IP encryption option, not required otherwise

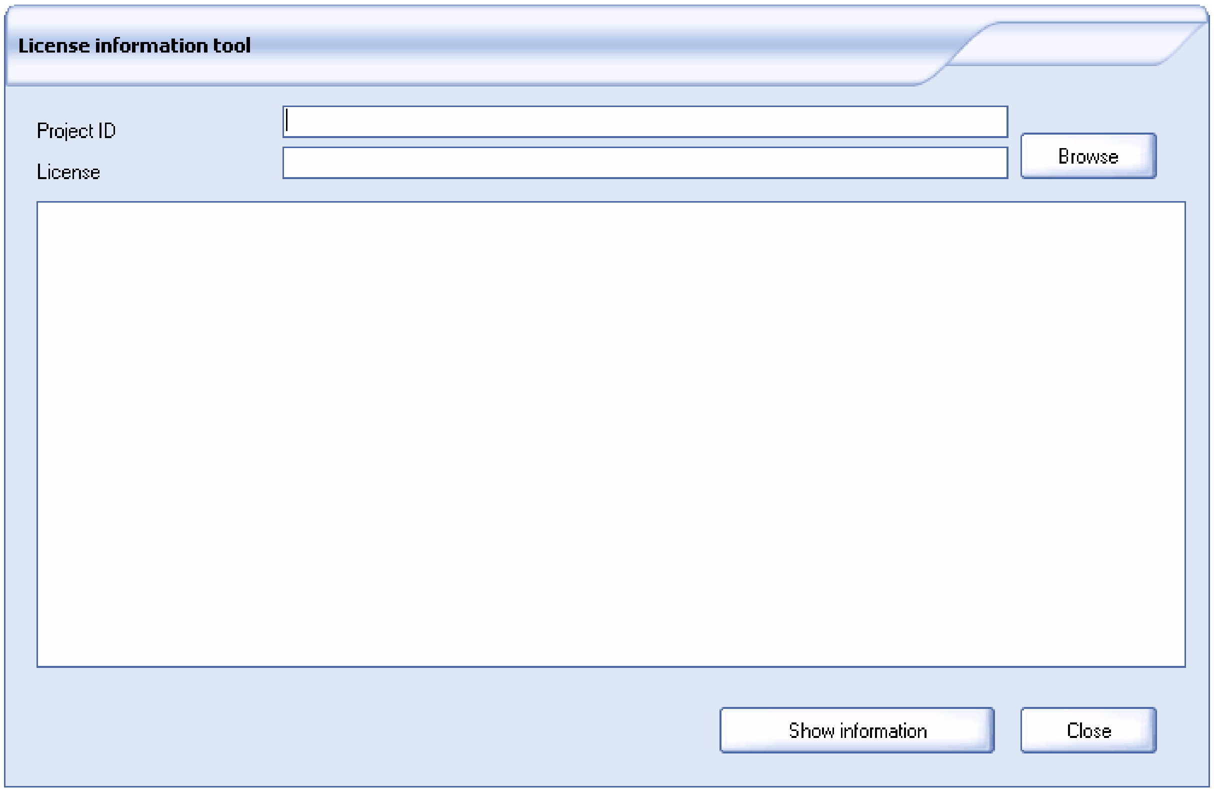
Target domain for decryption - required if script was encoded using domain encryption option, not required otherwise

Project Key - you may need to specify it if it was filled on encoding stage

Script - use 'Browse' button, or type directly script file name to specify script to analyse

Click on 'Show information' button and see information window for details.

License information tool



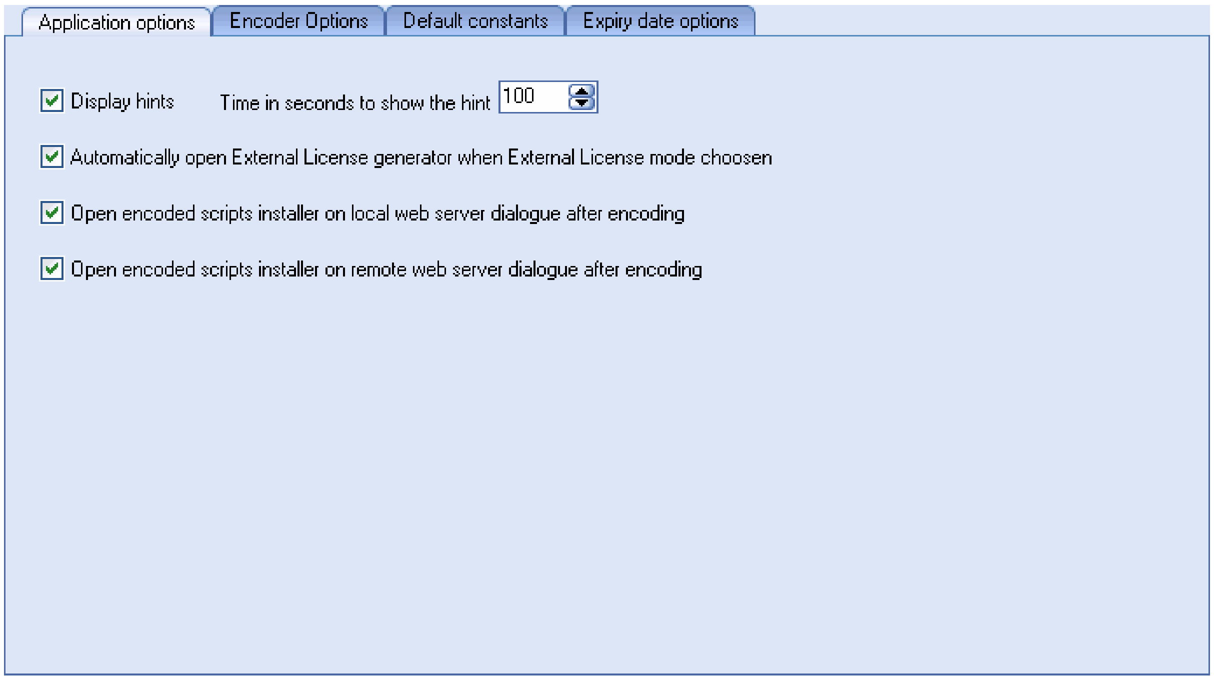
Project ID - you need to specify Project ID (see License generator section about this property)

License - use 'Browse' button, or type directly license file name to specify license to analyse

Click on 'Show information' button and see information window for details.

2.10 Settings

Application options



☑ Display hints   Time in seconds to show the hint 100

☑ Automatically open External License generator when External License mode choosen

☑ Open encoded scripts installer on local web server dialogue after encoding

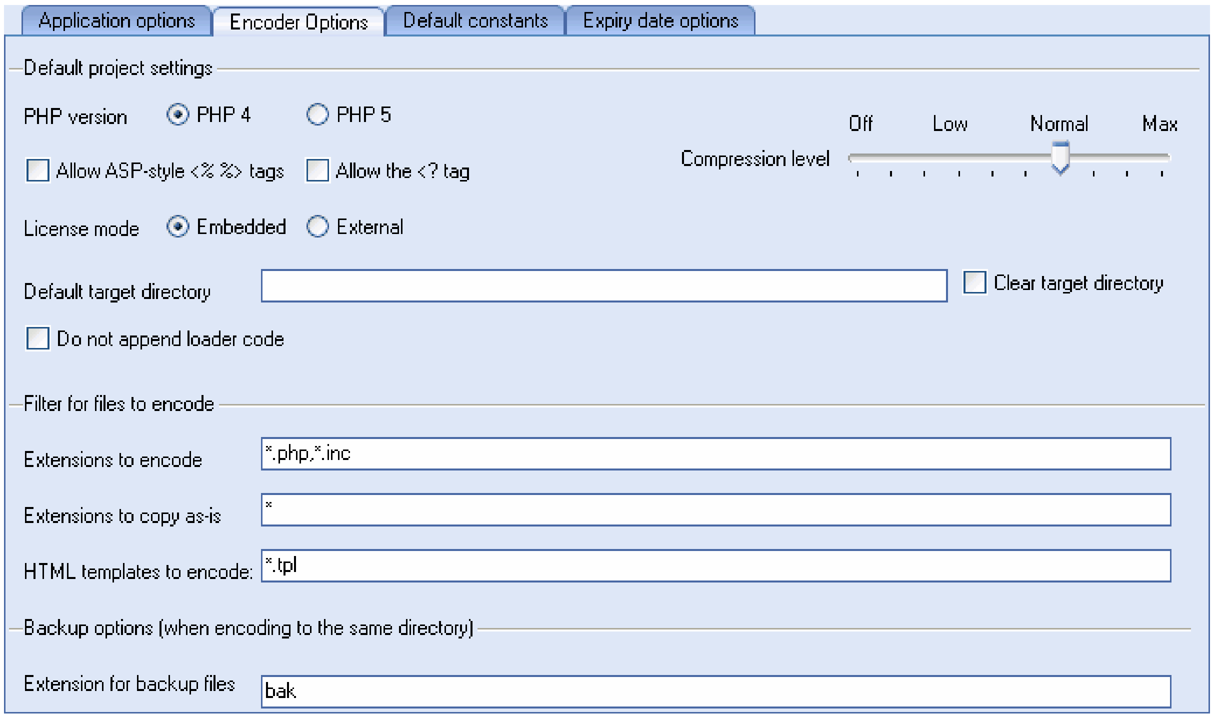
☑ Open encoded scripts installer on remote web server dialogue after encoding

Display hints - you can switch this ON/OFF. Displaying hints is useful when first using the program, when you are not familiar with all with the application options.

Time in seconds to show the hint - adjust this to your needs so you can read the tip.

Automatically open External License generator when External License mode selected - select this if you want automatically open External License generator after encoding your project when you choose External License binding mode recommendedd).

Encoder options



Default project settings

PHP version [✓ PHP 4] PHP 5

Allow ASP-style <% %> tags ☐ Allow the <? tag ☐

Compression level: Off — Low — Normal — Max (set at Normal)

License mode [✓ Embedded] External

Default target directory:

Clear target directory ☐

Do not append loader code ☐

Filter for files to encode

Extensions to encode: *.php,*.inc

Extensions to copy as-is: *

HTML templates to encode: *.tpl

Backup options (when encoding to the same directory)

Extension for backup files: bak

Default project settings

This allows you to set up default project settings so that when you start the New Project then these settings will be the default for your projects.

Filter for files to encode

Compression level [0-9]- here you can specify compression level for encoded scripts. Higher compression level gives smaller output scripts which run faster but encoding process will be slower (and vice versa).

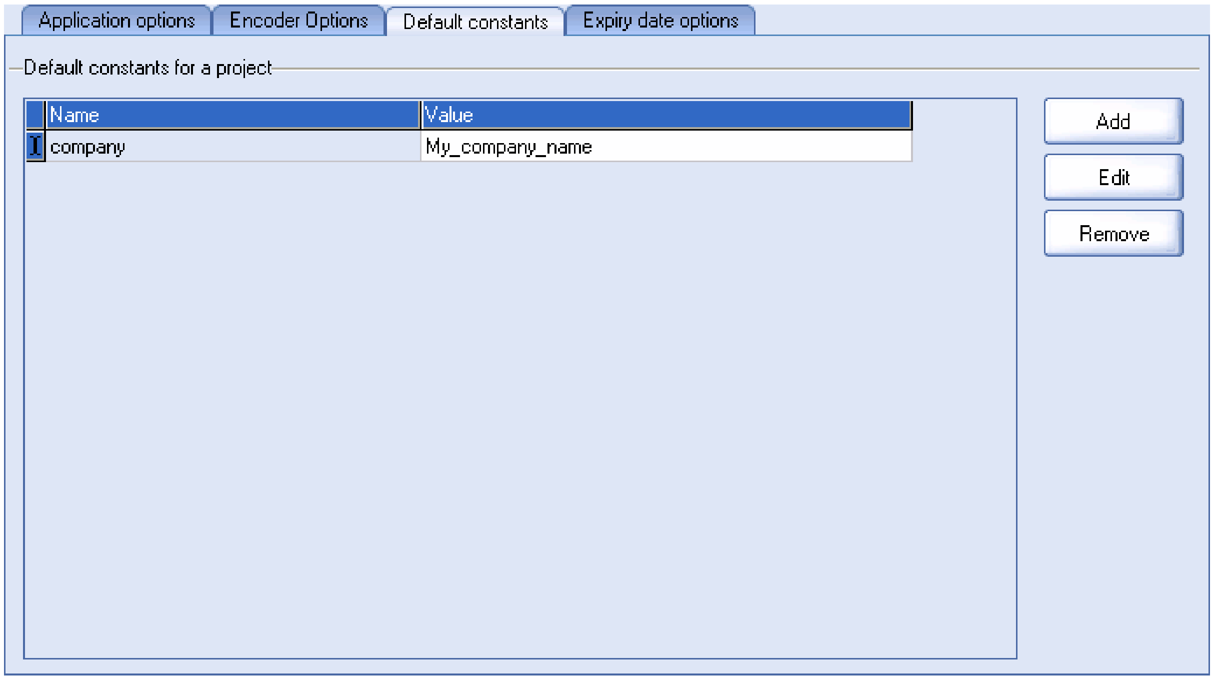
Extensions to encode - here you can list an extension that you wish to encode when you add files/directories to your project. You should separate extensions by commas. For example: *.php,*.php4,*.inc

Extensions to copy as-is - here you can list extensions that will be marked to copy as-is (ie without any encoding) when you add files/directories to your project. You should separate extensions by commas. For example: *.html,*.tpl,*.txt

Backup options

This option is used when you want to encode files in the same place (with overwrite of source scripts). For example: bak

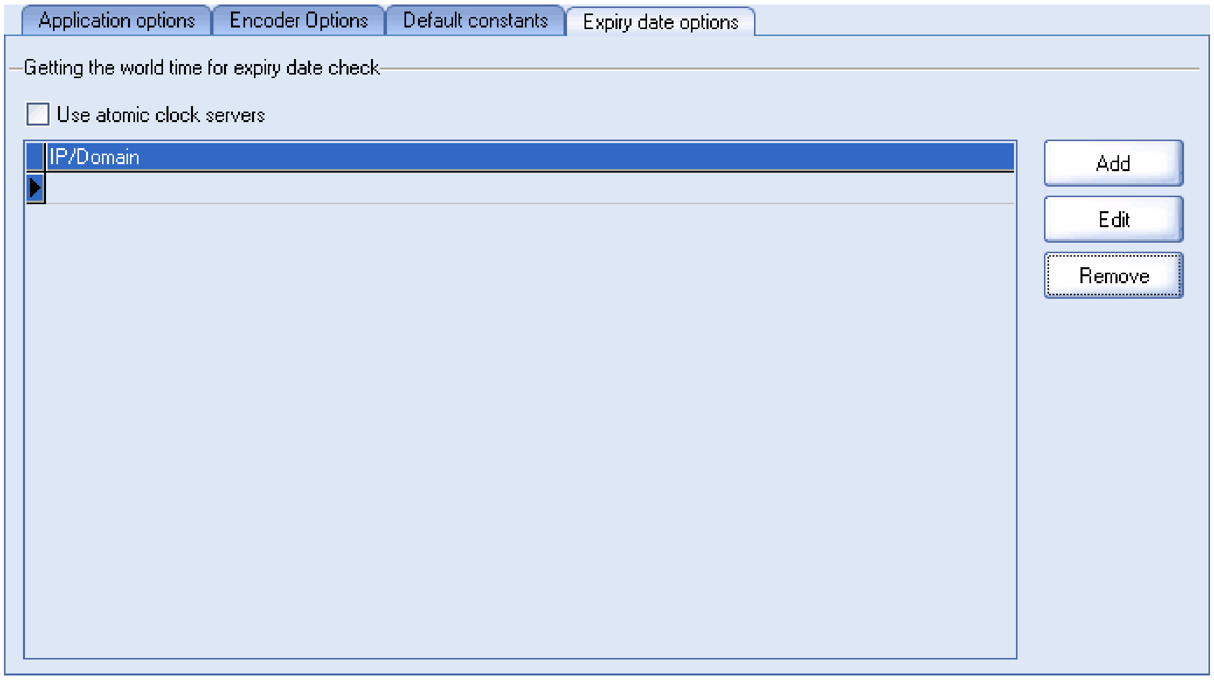
Default constants



Name	Value
company	My_company_name

Here you can specify default constants that will be added to custom constants for every your project. For example it may be your company name, etc.

Expiry date options



Use atomic clock servers - check this if you want to use atomic clock servers by default for every your new project.

IP/Domain - list of atomic clock servers that will be used for checking current time. The list of available time servers may be found here.

Part III

3 Protected script loaders (ixed loaders)

Scripts protected with SourceGuardian™ will require the installation of a SourceGuardian™ loader on the target
machine in order to run. Protected scripts will automatically attempt to find the loader in the ixed/ directory located within the protected script's directory or parent directories. SourceGuardian™ loaders may also be installed into a php.ini configuration file - This is useful, for example, if automatic loading is not supported or if faster performance is required.

For PHP versions 5.2.5+, SourceGuardian™ loaders need to be installed in the PHP extensions directory (extension_dir). You may find the extension_dir path in the php.ini configuration file or in the phpinfo() output.
The way the dynamic loading dl() function works in PHP has been changed since version 5.2.5 - It may load PHP extensions located **ONLY** in the extension_dir directory or a subdirectory within it. This means that SourceGuardian™ loaders cannot be loaded automatically from the ixed/ directory located within the protected script's directory or parent directories for PHP 5.2.5+. Usually you will get the following error message in that case: "Warning: dl() [function.dl]: Temporary module name should contain only filename". Please also read the note below about installing the loader for PHP 5.2.5+.

Protected Script Loaders will be updated periodically and the latest loaders are always freely available from:

http://www.sourceguardian.com/ixeds/

3.1 How to install the loader helper

We can help you to know how to install the loader on your destination system. Please look here

3.2 Loader filename structure

The following provides an overview of the loader naming conventions:

ixed.X.Y.Zdd.os

X.Y - major PHP version number (4.3 for 4.3.x, 5.0 for 5.0.x, 5.1 for 5.1.x, 5.2 for 5.2.x)
Z - minor PHP version number (2 for 5.0.2) This is optional and most loaders DOES NOT have the minor version number in the file name.

This part may be missed in the loader name which means that this loader is for all higher PHP versions:
ixed.4.3.lin - for all PHP 4.3.x versions
ixed.5.0.0.lin - for PHP 5.0.0 only
ixed.5.0.1.lin - for PHP 5.0.1 only
ixed.5.0.2.lin - for PHP 5.0.2 only
ixed.5.0.lin - for all PHP 5.0.3+ versions and higher
ixed.5.1.lin - for all PHP 5.1 versions and higher
ixed.5.2.lin - for all PHP 5.2 versions and higher

dd - optional code of supported encoder and system:
(missed) - this loader will load scripts protected with full version of SourceGuardian™
ev. - this loader will load scripts protected only with the evaluation version of SourceGuardian™
ts. - this loader will load scripts protected with the full version of SourceGuardian™ and is suited for manual installation on systems with Thread Safety enabled PHP (see details above)

os - three char code of operating system type. Currently supported:
.win - Windows
.lin - Linux
.fre - FreeBSD
.net - NetBSD
.ope - OpenBSD
.sun - SunOS
.dar - MacOSX (darwin)
.hp- - HPUX PA-RISC system

For some operating systems there are different versions of loaders for 32-bit and 64-bit mode. File names of such loaders are the same as it is impossible to determine 32/64-bit mode on a PHP level. Although 32-bit and 64-bit loaders are packed in different zip (tar.gz, tar.bz2) files so you can easily determine them. You need to use the correct 32-bit or 64-bit version of the loader on your system according to the platform and the mode PHP executable or shared object is built. You **may safely try** 32-bit version and then 64-bit one if you are unsure. Usually you will get the following error message in the case of wrong 32/64-bit loader is installed: "Unable to load dynamic library 'ixed....' cannot open shared object file" or "Unable to load dynamic library 'ixed...' wrong ELF class: ELFCLASS32(64)". If you have access to a command line shell you may check the PHP using command line "file" tool, e.g. "file /path/to/php".

3.3 Automatic ixed loading

Many operating systems and installations of PHP will load the Protected scripts without any modification. PHP will be able to find and load the appropriate loader if the following conditions are met:
1) Operating system and PHP mode:
Linux, FreeBSD, NetBSD, OpenBSD, MacOSX, SunOS, other UNIX - PHP installed as CGI or CLI
Linux, FreeBSD, NetBSD, OpenBSD, MacOSX, SunOS, other UNIX - PHP installed as a webserver's module (with thread safety off)
Windows - PHP installed as CGI or CLI
2) Thread Safety is disabled. You may check phpinfo() output for this.
3) dl() is enabled. You should have enable_dl=On in your php.ini.
4) The PHP extensions directory (extension_dir) needs to exist. Please check that the extension_dir= option in php.ini points to the real directory. Some hosting companies have incorrect installations of PHP and this can
cause problems.
5) The latest ixed loaders are installed in an ixed/ subdirectory within your scripts directory or any parent directory.
6) PHP version is below 5.2.5.

Please note: if your server and PHP configuration conform to all conditions above for automatic loading except only a PHP version, then it is enough to copy an appropriate loader into the PHP extension directory (extension_dir). The loader will be used automatically from the extension_dir directory - no need for changes in the php.ini configuration file.

3.4 Manual ixed installation

It is possible to manually install the loader and this is required in the following conditions:
1) Operating system and PHP mode:
Linux, FreeBSD, NetBSD, OpenBSD, MacOSX, SunOS, other UNIX - PHP installed as a webserver's module (with thread safety on)
Windows - PHP installed as Apache module (thread safety is always on)
2) If Thread Safety is enabled. You may check phpinfo() output for this. PHP installed as a

webserver's module under Windows will always have Thread Safety on.
3) If dl() is disabled. You have an "enable_dl=Off" setting in the php.ini configuration file.

Manual installation requires permissions to access the extension_dir directory and the php.ini configuration file. Manual installation may be used even if automatic loading is available. With appropriate and manually
installed SourceGuardian™ loaders you give the maximum performance for your protected scripts. This is because the script will not need to search for a loader each time it runs.

To install the SourceGuardian™ loader manually you need to do the following:
1) Choose an appropriate loader for your operating system and version of PHP. Please refer to the "Loader filename structure" section below to know which loader is required for your operating system and version of PHP.
2) Find the loader file in the ixed/ subdirectory within SourceGuardian main installation directory and copy it to the PHP extension directory (extension_dir - check the phpinfo() output).
3) Find the location of the php.ini configuration file (check the phpinfo() output) and add "extension=ixed.X.X.YYY" directive at the end of the file. (X.X is the major version of PHP and YYY is the name of operating system)
4) Restart the webserver in order to apply changes done in the php.ini configuration file and reload PHP.
5) Optionally you may check the phpinfo() output now to find out that SourceGuardian loader was successfully installed - search for "SourceGuardian" within the output.

3.5 Zend extension support

Since version 5.0 SourceGuardian loaders may be loaded as Zend extensions. This allow you to specify the full absolute path to the loader regardless of the extension_dir setting. Of course the PHP or webserver process should have enough permissions to access the loader in that location.

To install loader for *non threaded* PHP use zend_extension option in php.ini:

```
zend_extension = /usr/local/ixed/ixed.4.3.lin
```

For *threaded* PHP use zend_extension_ts option in php.ini: (mod_php apache module is always threaded under windows)

```
zend_extension_ts = /usr/local/ixed/ixed.4.3.lin
```

Also you should specify an appropriate loader for your OS and PHP version. See the "server wide install" section in our user's manual about loaders names.

3.6 Execute only SourceGuardian protected scripts

It's possible to setup PHP to execute only SourceGuardian protected scripts. The SourceGuardian loader should be installed *server-wide* in php.ini and then the following option is set in php.ini:

```
[SourceGuardian]
sourceguardian.restrict_unencoded = "1"
```

If an unencoded script is executed the following error message appears:

```
Fatal error: SourceGuardian Loader - unencoded script cannot be executed [08]
```

Part IV

4 Command line encoder

4.1 Ultimate PHP Scripts Protection

The SourceGuardian™ 7.1 for PHP Encoder protects PHP scripts by compiling PHP source code into a bytecode format and this is followed by encryption. This protects your scripts from reverse engineering.

To protect your scripts from unauthorised usage SourceGuardian™ 7.1 for PHP has added features that can optionally lock your scripts to run only from predefined IP addresses, domain names or LAN hardware addresses (MAC). SourceGuardian™ 7.1 for PHP can also easily produce trial versions of your scripts by setting an expiry date for the script or by limiting the number of days that protected script will work. For larger projects SourceGuardian™ 7.1 for PHP provides an option to protect an entire project so that all scripts used in the project will work only with other protected scripts. No one may include a protected script from another unprotected script and this adds another level of protection.

4.2 Licensing your protected scripts

With SourceGuardian™ 7.1 for PHP you may optionally lock your scripts so that they require a special license file in order to run. This file may be distributed with the script or separately from it and this option gives you an opportunity to encode your script once and distribute to users with different licenses. Each license may have different and specific attributes.

4.3 Supported PHP versions

SourceGuardian™ 7.1 for PHP supports encoding for PHP 4.1.x, 4.2.x, 4.3.x, 4.4.x and PHP 5.x.x. There are two different executable files for PHP 4 and PHP 5 encoding: encode4 and encode5. Scripts encoded for PHP 4 and PHP 5 have a different internal format and are not replaceable with each other - This provides added protection and ensures compatibility. Any scripts encoded with encode4 will not run on a PHP 5 system and vice versa.

4.4 Cross platform encryption

SourceGuardian™ 7.1 for PHP protected scripts have the same internal format for all supported operating systems. This mean that scripts encoded with SourceGuardian™ 7.1 for PHP under a Windows OS will also run under a Linux OS with the appropriate linux loaders. This flexibility is the same for all other supported operating systems.

4.5 Command line encoder installation under Linux

The SourceGuardian™ 7.1 for PHP installation package is a .tar.gz file called:

Full version: sg4full_linux.tar.gz
Trial Version: sg4eval_linux.tar.gz.

You need to unpack this file into any directory you wish.

Example:

```
> cd /usr/local
> mkdir sg4
> cd sg4
> tar xzf /path/to/sg4full_linux.tar.gz
```

You may be need to add permissions under Linux to allow to the executing of the encoder binaries:

```
> chmod u+x encode4 encode5 licgen
or
> chmod a+x encode4 encode5 licgen
```

The installation package has following structure:

bin/encode4	(encoder for PHP 4)
bin/encode5	(encoder for PHP 5)
bin/license.txt	(license text)
bin/licgen	(for full version only)
ixed/ixed.*	(protected script loaders)
README	(documentation about SourceGuardian™ encoder)

You may optionally add symlinks to encode4, encode5 and licgen if needed to any other directory.

4.6 Command line encoder installation under Windows

The command line encoder is already installed if you have installed SourceGuardian™ 7.1 for PHP under Windows. The command line tools are located in the \encoder\ subdirectory within the SourceGuardian™ 7.1 for PHP installation directory.

The Path for default installation of the full version is:
C:\Program Files\SourceGuardian\SourceGuardian 7.1 for PHP\encoder\

The Path for default installation of the evaluation version is:
C:\Program Files\SourceGuardian\SourceGuardian 7.1 for PHP demo\encoder\

4.7 Running the command line encoder

SourceGuardian™ 7.1 for PHP is a command line tool for PHP script protection. You should have an access to a console or any kind of remote shell to run it under Linux or FreeBSD. Although we have a GUI application for Windows you may prefer to use a command line in some cases, for example for automated encoding or license protection.

There are two different executable files for PHP 4 and PHP 5 encoding:
encode4 and encode5.

4.7.1 First run

Under Linux or FreeBSD you will have to read and accept the SourceGuardian™ 7.1 for PHP license during the first run of the encoder. The License will only be displayed on the first run. Please read it and, if you accept the agreement, press Enter/Return key for the next page. You need to accept the license terms to continue.

If the SourceGuardian™ 7.1 for PHP license is accepted you will get a web link to our site and a hexadecimal registration code on the screen. You need to visit the following URL:

http://www.sourceguardian.com/profile/

and login and enter this registration code on to your profile page to get a license for running the application. Download the license (encode.lic) that is created and copy it into the command line encoder installation directory.

4.7.2 Usage

single file:	encode5 [options] file.php
multiple files:	encode5 [options] file1.php file2.php file3.php
file mask:	encode5 [options] *.php
file list:	encode5 [options] @filelist

You may run the SourceGuardian™ 7.1 for PHP encoder to encode either one or multiple files. You may enumerate all files you want to encode or use a file mask or file list to specify multiple files. A file list is a text file with either full or relative file paths of all the files to encode, separated by a new line (masks are supported since 5.0, use '*' and '?' for it). You should use an @ sign before the filelist name in the command line.

A file list passed to the SourceGuardian encoder for batch processing from the command line may contain file masks. Standard ? and * symbols are available.

The encoded file will replace the original file. The original file will be backed up with a .bak extension by default (until you turn off the backup facility with a -b- option).

4.7.3 Options

The available options are:

-v	Display version number
-h	Display help with full options list
-l	Display license information
-w	Wait for key press before exit
-q	Display settings and request confirmation. Encoder will display all encoding parameters and wait for a key press before real encoding takes place. You may check all parameters and cancel if anything is not correct.
-r	Recurse subdirectories. The encoder will recurse all subdirectories when searching files using a specified file mask.
-b	Set file extension for backup files (bak is default). You may change an extension used for backup copies with this option. Example: -b old
-b-	Disable backup of source files (be careful!)

4.7.4 Script locking options (full version only)

--expire [dd/mm/yyyy]

With this option you can set an expiration date for the script. The script will not run on and after the specified date and comes with the error message: "script has expired". This option will override any previous lock set with the --days option.

--days [nn]

You can set the script to expire in a number of days (from today). The script will not run after nn days from today and comes with the error message: "script has expired". This option will override any previous lock set with --expire option.

Getting the world time for expiration date checking

If you use a time lock option for your scripts you may wish to let the script get the world time from the online time service for expiry checks rather than using the server time. You may specify a list of time services during encoding.

Use --time-server option to specify time servers. You may specify multiple servers IP addresses or domain names separated with "," or ";"

The "time" protocol is used, tcp/ip to port 37 on the server.

If you have used a time-server option then your script will *require* an internet connection to run. Time servers will be checked in the specified order. If no server from the list can be accessed an error message will be displayed and the script will stop execution:

"script requires an internet connection to run [20]"

It's a good idea to specify 2-3 time servers which will let your script to work even if some of the time servers will be temporary down.

If you have multiple scripts included from each other and some of them were encoded with a time-server option then the script will access the time server only once for the first script for better performance and will use the time value from the time-server for other included scripts.

The list of available time servers may be found here.

* Locking the script to work only online

You may also use the time-server option to lock your script to run only online. Use time-server option as usual for this but don't specify an expiration date for the script. The script will try to access the online time service and will fail if it's not possible.

--domain [domain]

You can bind the script to a domain name. The Encoder will lock the script to run only from the specified domain and all sub domains. If an attempt is made to run the script on a non-authorised domain, the following error message will be displayed: "script cannot run on this machine". You may use this option more than once to specify multiple domains. This option may not be used with the --domain-encrypt option.

Hint: use the name of the main domain in this option, not the name of any sub domain until you are sure you need to lock to a sub domain.

Example 1: --domain mydomain.com

The script will run from mydomain.com, www.mydomain.com, myname.mydomain.com etc but will NOT run from otherdomain.com, www.otherdomain.com, otherdomain.net etc.

Example 2: --domain www.mydomain.com

Script will run ONLY from www.mydomain.com. It will not run on the Main domain mydomain.com and all other sub domains like myname.mydomain.com as well as other domains like otherdomain.com, www.otherdomain.com, otherdomain.net etc.

--domain-encrypt [domain]

Bind and encrypt to domain name. The encoder will lock the script to run only from the specified domain. The encoder will use a specified domain name as a part of the key for encryption for the maximum protection. The loader will not be able even to decrypt a script from the wrong domain and will display the error message: "script checksum error". You may use this option ONLY ONCE in a command line. This option may not be used with the --domain option.

Be careful when using this option if you may possibly need to run your protected script from a sub domain. Example: --domain-encrypt mydomain.com will allow to run script ONLY from mydomain.com not even from www.mydomain.com and vice versa.

Domain name locking supports wildcards. You may lock to *.site.com and so the script (or external license) will work for aa.site.com, bb.site.com etc. ? and * symbols are supported in the same manner as for specifying file masks.

--ip [x.x.x.x{/y.y.y.y}]

Bind script to an ip/mask. The encoder will lock the script to run only from the specified IP address. The specified IP address mask will be applied to the real IP address before comparing. So you may use this option to lock the script to a multiple IP if mask is specified. If run from not allowed IP script will come with the error message: "script cannot run on this machine" You may use this option more than once to specify multiple ip/mask pairs. IP address mask 255.255.255.255 is used by default if not specified. This option may not be used with --ip-encrypt option.

--ip-encrypt [x.x.x.x{/y.y.y.y}]

Bind and encrypt to ip/mask. The encoder will lock the script to run only from the specified IP address. The encoder will use a specified IP address with applied mask as a part of the key for encryption for the maximum protection. The Loader will not be able to even decrypt a script from the wrong ip address and will display the error message: "script checksum error". You may use this option ONLY ONCE in a command line. IP address mask 255.255.255.255 is used by default if not specified. This option may not be used with --ip option.

--mac [x:x:x:x:x:x]

You can bind a script to LAN hardware (MAC) address. This address is unique for a networking adapter and so it may be easily used to identify a machine. A MAC address is 6 bytes long, with each byte represented in hex and separated with ':' The encoder will lock a script to run only from the machine which has a networking adapter with the specified MAC address. If there is more than one LAN adapter installed then script will check all of them. If an attempt is made to run a script from a machine without the correct adapter, then the script will display the error message: "script cannot run on this machine" You may use this option more than once to specify multiple MAC addresses.

Hint: you may use 'ifconfig' command under Linux or 'route print' under Windows to get a list of installed networking adapters and known MAC addresses.

--external [filename]

Script will require external license file to run. This file may be distributed with the script or separately from it. This option gives you an opportunity to encode your script once and distribute to users with different licenses. Each license may have a different number of locks. You should specify only an external license file name here. Example: --external script.lic No real license file will be created for now. You should use licgen tool for creating a license file for the script or you may do it via GUI if under Windows. When running protected scripts, and no specified license file is found, the script will come with the error message: "script requires ... file to run" You may use this option only ONCE in a command line. This option may not be used with any other binding options.

--projid

Allows you to specify Project ID to identify your project. To be used with --external option. You should use licgen tool for creating a license file for the script with the same Project ID.

--conj

Script will work only with other encoded files. This option makes sense only when encoding multiple files. All scripts encoded with this option will work only with other encoded files and will NOT work if any of the included files or top files are substituted with an unencoded one or encoded by another installation of SourceGuardian™ for PHP. This gives you the ultimate protection for your projects when multiple PHP scripts are used together.

Example: If you have a password in a.php and then b.php includes a.php and calls c.php for any action. No one can substitute c.php with their own code and do 'echo $password' to know your password if this option was used during the encoding of all files. Also no one can create d.php which will include protected a.php and then do 'echo $password'.

NOTE: Since SourceGuardian 5.0 this option was changed to allow including and executing only scripts from the same project (with the same Project Id value). This lets you develop and encode parts of your project on multiple machines (with multiple SourceGuardian licenses) and keep the "conjunction" option on for maximum protection.

We recommend to always use this feature if your project has any secure data embedded in scripts such as usernames, password, database names etc.

Since SourceGuardian 5.0 the "conjunction" option is always applied to the script during encoding and not to the external script license.

4.7.5 Advanced options

--asp-tags

Enables the use of ASP-like <% %> tags in source code. Use this option if you use asp-style tags in your PHP scripts.

--short-tags

Enables the use of short PHP's <? ?> tags in source code. Use this option if you use short PHP's tags in your PHP scripts.

-p "code"

Prepend header code. You may put any code to be executed BEFORE the protected scripts code. This code WILL NOT BE ENCODED. This may be either HTML text or PHP code. For PHP code - you should use <?php ?> tags. This option is usually used for including copyrights into protected scripts. You should prepend all double quote characters with a back slash if you want to include them into the code (" -> \").

Example 1:

-p "<!-- My protected script. Copyright by \"My Name\" -->"

Example 2:

-p "<span class=\"bold\">My protected script. Copyright by My Name</span>"

Example 3:

-p "<?php echo \"My protected script. Copyright by My Name\"; ?>"

-j "code"

It is possible for you to change the default loader error code. This option allows you to change the default error action of the protected script if it cannot find an appropriated ixed loader. The default action is just to print an error message "This script is protected by SourceGuardian™ and requires file ... " and stop executing. You may use any HTML text or PHP code here and it will be displayed or executed as a replacement to the default SourceGuardian™ loader error. This code WILL NOT BE ENCODED. This may be either HTML text or PHP code. For PHP code - you should use <?php ?> tags. You should prepend all double quote characters with a back slash if you want to include them into the code (" -> \").

Example 1:

-j "<a href=\"email:admin@domain.com\">Contact administrator</a>"

Example 2:

-j "<?php header(\"Location: /myhandler.php\"); exit(); ?>"

<u>Prepend header code and Loader error code may be loaded from a file</u>

Prepend header code option -p and Loader error code option -j may load the source from a file. Use @filename as a parameter for -p or -j.

Example: encode4 -p @prepend.php -j @loadererr.php myscript.php

The code is loaded during encoding and stored as is *non encoded* as that code should be executable when no ixed loader is loaded yet.

-n

Don't integrate the ixed loader. you may use this option if you don't want to include a default loader into the protected script. Any script encoded using this option will not be able to automatically find and load the appropriate ixed loader and you have to install the ixed loader manually to run this script. See the section above about the manual ixed installation. This option may not used with -j option.

-z[0-9] - compression level. Higher compression level gives smaller output scripts which run faster but encoding process will be slower (and vice versa).

4.7.6 Excluding files by the file mask

You may exclude some files or directories from encoding when using the command line encoder. Use --exclude=mask option to specify file(s) and/or dir(s) to exclude from processing. You may specify either a strict name, relative path with a directory name or a mask (with ? and/or *).

UNIX users: Always quote masks under UNIX. Otherwise the shell will replace your mask with real file and dir names and the result may be unexpected. You should always quote masks that specifies files to encode too (like "*.php" in the example below).

Example: encode4 -r --exclude "doc/*" --exclude "config.php" "*.php"

This will encode all *.php files in the current directory and all directories recursively but all files in the "doc" directory and all files (and dirs if any!) named "config.php" will not be encoded.

You may enumerate all the files you want to exclude from encoding using a file list to specify multiple files. A file list is a text file with either full or relative file paths of all the files to encode, separated by a new line (masks are supported, use '*' and '?' for it). You should use an @ sign before the filelist name in the command line. Usage: -x @filelistname

4.7.7 Excluding files from encoding but still copying to output directory

We have added an option into the command line encoder to specify which files should be encoded (-f). You may specify what files will be encoded by filenames, filemasks or filelist. All other files which have been added for processing or found by expanding filemasks will be copied into the output directory "as-is" without encoding. If you don't specify the -f option then all specified files will be encoded by default.

Example 1:
>encode5 -r -f ".php" -o "output_dir" "*"*
All (with recursion) *.php files from the current directory will be encoded and copied into the output_dir. All other files from the current directory will be copied into output_dir as-is (unencoded).

You may specify multiple filenames or filemasks with using of multiple -f options:

Example 2:
>encode5 -r -f ".php" -f "includes/*.inc" -f @myphpfiles -o "output_dir" "*"*

If you don't specify the output directory but use -f option then only files specified with -f option will be encoded. All other files will remain unchanged.

4.7.8 Encoding directory content without specifying filemasks

It's possible to use shorter syntax for directory encoding. All specified directories will be recognized and the "*" filemask will be added:
>encode5 -r source_dir

instead of the following in previous versions:
>encode5 -r "source_dir/"*

4.7.9 Output directory for encoded scripts

You can specify an output directory for all encoded scripts when encoding from command line. Source files will be unchanged if you specify an output dir different from your source scripts dir. The default backup option will be off when an output dir is specified. If you want to re-enable it, even when the output dir is specified, then use the -b <backup_extension> option after the output directory option.

The full directory path to source scripts will be recreated under the output directory if the full path to source files was specified. Windows users - drive names ("C:","D:",etc) will be replaced with just one letter ("C","D",etc) when recreating the path under the output directory.

Command line option: -o <output_dir>

Example 1: Encode all *.php scripts in the current dir with recursion and put encoded files into /home/myproject/encoded.

```
> encode4 -r -o /home/myproject/encoded *.php
```

Example 2: Encode all scripts specified in the filelist and put encoded files into /home/myproject/encoded. Additionally backup source scripts in source directory with .bak extension.

```
> encode4 -o /home/myproject/encoded -b bak @filelist
```

4.8 Additional bytecode protection

Since version 5.5 of SourceGuardian has new options for stronger bytecode protection. In additon to our standard bytecode compiling, encrypting and compressing, the encoder can obfuscate the names of variables, functions and classes within the bytecode to make it an unreadable value.

Additional bytecode protection may be set from levels 1 to 3. For levels 2 and 3 it will require a greater knowledge of your scripts and will require some manual intervention to exclude some variables, functions or classes from names changing - this may be necessary to ensure your application works as it should do.

If no bytecode protection level has been selected, then the encoder will work without obfuscation, but your scripts will still be compiled into bytecode, encrypted and compressed.

4.8.1 Levels of bytecode protection

Each level of bytecode protection includes the functionality of the previous one, but also adds more protection.

Command line option: --prot <level>

Level 1 obfuscation - This provides name changing for local function variables. This level is safe to use as local function variables cannot be accessed externally and so may be easily renamed by SourceGuardian.

RECOMMENDATION: Use this option to quickly add additional protection when you don't want to analyze your code for adding to the exclusion lists.

Level 2 obfuscation - This level additionally changes the names of all global vars, function names and class names defined within the file.

RECOMMENDATION:

1) if register_globals is used and EGPCS vars are used directly then their names should not be changed. You need to put such global variables names into the global variables exclusion list (see below)

2) global variables accessed via the $GLOBALS array by name should keep their names and so need to be put into global vars exclusion list. By default names will be changed for all global variables and they may still be accessed correctly if within the project even if they are in different files.

3) global variables used in the project and accessed externally from other unencoded (or encoded with other options) scripts should keep their names and they should not be obfuscated so they need to be put into the global vars exclusion list.

4) possible errors could occur if a function or a class is defined in any script that is used in another script. If so put this function name or class name into the appropriate exclusion list.

Example for this level only:

```
a.php: <?php function myfunc() { echo 'test'; } ?>
b.php: <?php include "a.php"; myfunc(); ?>
```

The myfunc() function above is defined in one file and is also used in another one. To run under level 2 you need to put the "myfunc" function name in to the functions exclusion list.

Use this option for encoding separate scripts or projects which have not much code interaction between separate scripts. Each function or class defined in one script and used in another script should not change its name and so needs to be put in the exclusion list.

Level 3 obfuscation - additionally this changes the names of all functions, classes, class methods and class properties.

RECOMMENDATION:

Same as for level 2:

1) if register_globals is used and EGPCS vars is used directly then their names should not be changed. You need to put such global variables names into the global variables exclude list (see below)

2) global variables are accessed via the $GLOBALS array by name should keep their names and so to be put into global vars exclusion list.

3) global variables used in the project and accessed externally from other unencoded (or encoded with other options) scripts should keep their names so need to be put into the global vars exclusion list. By default names will be changed for all global variables and they still may be accessed OK if within the project even in different files.

New to level 3:

4) names of functions and classes defined in the extensions should not be changed and so need to be defined in the exclusion lists. ex.
   mysql_connect(), mysql_select_db(), mysql_query()...(mysql),
   preg_match()...(pcre), MimeMessage class (mailparse) etc.

This option will change the names for the user defined or extended functions and classes. It will not change the names for standard built-in functions such as print(), str_replace(), etc and built-in classes such as Exception, Iterator etc.

One other difference from level 2 is that as names will be changed for all functions and classes by default then there is no problem for the script to include a file from the same project and call a function defined in the included file or to create an object of the class defined there.

Example for this level only:

```
a.php: <?php function myfunc($arg) { return preg_match('/test/',$arg); } ?>
b.php: <?php include "a.php"; myfunc(); ?>
```

preg_match() is not a standard PHP function. This function is defined in separate pcre extension and so its name "preg_match" should be put into the functions exclusion list to run under level 3. Please note, there is no problem with the
myfunc() function being defined in one file and used in another one under level 3.

Use this option for encoding the whole projects which will include everything needed to run your application and which has no unencoded functions or classes. There are no problems to still have a custom settings file as such settings are usually done via assigning global vars, constants or even ini files.

RECOMMENDATION for all levels of bytecode protection; Except for level 1 the encoder does significant changes to the bytecode this creates additional protection. You should test your encoded projects well and understand the risk of this kind of additional protection that if not used correctly you may have obfuscated and unobfuscated information that will then not work well together. Analysis is definitely required for using levels 2 and 3 and this cannot be done automatically by the encoder. Therefore you need to fill in the exclusion lists for global variables, functions, classes and class properties according to your code.

4.8.2 Bytecode protection exclusion lists

For bytecode protection levels 2 and 3 you need to specify the names of global variables, functions, classes as well as class properties for which names should not be changed. Use the following options to do it.

* Global variables exclusions:

Command line option: -OG<var_name1> -OG<var_name2> ...
or from the filelist: -OG@<filelist> (one name per line)

* Function/class methods exclusions:

Command line option: -OF<func_name1> -OG<func_name2> ...
or from the filelist: -OF@<filelist> (one name per line)

* Classes exclusions:

Command line option: -OC<class_name1> -OG<class_name2> ...
or from the filelist: -OC@<filelist> (one name per line)

* Class properties exclusions:

Command line option: -OP<prop_name1> -OG<prop_name2> ...
or from the filelist: -OP@<filelist> (one name per line)

4.9 Using external script license generator (full version only)

The Script License Generator is an external tool for creating script license files. A Script license file is required to run protected scripts encoded with the --external option.

Using the script license is the best way of encoding if you need to distribute one script or entire project between different users but need to use different restriction options for each user. You need to encode your scripts with the --external option using SourceGuardian™ 7.1 for PHP and then create a license for each user with the SourceGuardian™ 7.1 for PHP Script License Generator.

Scripts encoded with the --external option will require an external license file to run. Protected scripts will search for the license file in the current directory and all parent directories. So you may have one license file for an entire protected project located in the top project directory.

If the protected script cannot find the specified license file it will come with the error message: "script requires ... file to run"

A new protection algorithm for external script license protection has been introduced in SourceGuardian 5.0. This algorithm gives your scripts much stronger protection from reverse engineering, unlocking and bytecode stealing, but it also gives you the most flexible way to generate trial versions of your products and to lock scripts to your customer's machine.

This method requires the use of a *license file* for your protected scripts to run. This is most powerful and flexible way to protect your scripts. We recommend you to use external license files for all your script protection.

Algorithm description

This new algorithm uses the idea of two keys. The first key (Project Id) is stored in the encrypted area of the protected script and is used to decrypt an external license file. The second key (Project Key) is stored in the license file and it is used to decrypt the bytecode from the protected script.

Using the new algorithm protects your product by preventing a full working copy from being created from, for example, a demo version. To decrypt and run a protected script a true license file for the full version of your product is required. Otherwise it's impossible to decrypt and run the bytecode.

Project Id and Project Key values are required if the external license protection method is chosen.

You should specify Project Id (--projid) and Project Key (--projkey) values using options in the command line for "encode4" or "encode5" commands. Project Id and Project Key may be any words, numbers or random sequence but for security reasons these two values *should not* be calculated from each other. They should be independent. Also you should specify the *same* Project Id, Project Key pair for "licgen" command when generating a license for previously protected scripts.

Command line example:

```
encode4 --external script.lic --projid "19Gh42Ki" --projkey "Ab65qZ32" myscript.php
licgen  --projid "19Gh42Ki" --projkey "Ab65qZ32" --days 7 script.lic
```

- *Other advantages*

If you have licenses for multiple SourceGuardian installations you may encode scripts on one machine and generate license files on another machine. The only condition is to set the same Project Id and Project Key values for your project on different machines.

- *Error messages*

If a script is run with an incorrect license file the following error message will appear:

"Fatal error: SourceGuardian Loader - script license is invalid [06] in ..."

If script is run with a license file with correct Project Id but incorrect Project Key (this may be a cracking attempt or accidental modification of the license file or script) the following error message will appear:

"Fatal error: SourceGuardian Loader - script checksum error [12] in ..."

- ***Important Security Notice!***

(!) Keep your Project Id and Project Key values in a secret.

(!) Remember your Project Id and Project Key. It's impossible to restore the values from somewhere if forgotten. They are required for generating licenses for your customers. GUI users - Project Id and Project Key are stored in SourceGuardian project file.

(!) When generating the Project Id and Project Key manually, please use independent values.

- *Encoding without external script license file*

You may still encode your scripts with an embedded license. In this case Project Id and Project Key values are not required. Project Id will be required for "conjunction" feature (see below).

4.9.1 Usage

licgen [options] output.lic

output.lic - this is the name of license file to generate. It should be same that you used in --external option during encode.

Options

-v   Display version number
-h   Display help with full options list
-l   Display license information
-w   Wait for key press before exit

4.9.2 Script locking options

All options listed below work exactly the same as the binding options of SourceGuardian™ 7.1 for PHP. Please refer to the "Script locking options" section above for details.

--expire [dd/mm/yyyy]	Set script expiration date
--days [nn]	Set script expiration days (from today)
--domain [domain]	Bind script to domain name
--ip [x.x.x.x{/y.y.y.y}]	Bind script to ip/mask
--mac [x:x:x:x:x:x]	Bind script to mac address
--projid	Set Project ID
--projkey	Set Project Key

4.10 PHP shell scripts encoding

SourceGuardian will keep the first line unchanged of the script if it begins with a #! UNIX shell script prefix (ex. #!/usr/bin/php) This lets protected scripts run from the shell. The first line of the script will not be encoded but the whole script including this line will be still protected with a checksum and so remains protected from unauthorized modifications.

4.11 Custom predefined constants

SourceGuardian also lets you define custom named constants during an encoding process, or within an external script license. Constant name/value pairs are stored internally in the encrypted area of the protected script or external license. They may be used for custom script locking or any other actions if you need to store a custom value in the protected script or script license file and then retrieve it from your PHP code.

Use --const name=value option in encode4, encode5 or licgen command. Use quotes if your constant name or its value contains any spaces or other special symbols:

```
encode4 --const "licensed_for=Robin Hood" myscript.php
licgen  --const "licensed_for=Robin Hood" script.lic
```

You may define only one constant with each --const option but you may add as many --const options as you need into the command line.

To get a predefined constant value from the PHP code use sg_get_const() function. This function is defined in the SourceGuardian loader.

Syntax: string sg_get_const(string)

> Will return a predefined SourceGuardian constant value or FALSE if constant with the specified name is not defined. SourceGuardian constants names are *case sensitive*.

There are 5 constants predefined for each protected script:

sg_get_const("encoder")	Returns the name of encoder "SourceGuardian"
sg_get_const("version")	Returns the encoder version number
sg_get_const("encode_date")	Returns UNIX timestamp when the script was encoded
sg_get_const("license_date")	Returns UNIX timestamp when the script license was created. It's may differ from "encode_date" when external script license is used
sg_get_const("expire_date")	Returns script expiration date as UNIX timestamp if it's defined in the script license or internally via script binding options during encoding

4.12 Custom error handling

You may add custom error handling functions which will catch script licensing errors. Error handler should be a function which accepts two parameters:

sg_error_handler(code , message)

You may use any name for this function. Also you may have different functions for different script errors. The first argument will contain an error code. The second one will contain a default error message.

To set a custom error handler use --catch option in encode4 or encode5 command:

```
encode4 --catch err=function myscript.php
```

Where "err" is one of predefined constants and "function" is error handler function name.

Err	Code	Default message
ERR_LICENSE	01,02,03	script cannot run on this machine
ERR_EXTLICCRC	06	script license is invalid
ERR_EXPIRED	09	script has expired
ERR_EXTLIC	13	script requires license file to run
ERR_OFFLINE	20	script requires an internet connection to run
ERR_ALL	-	-

ERR_ALL is a special value to specify "one-for-all" error handler function.

Custom error handler function should be defined before an error may occur. The best place for it is in "prepend header" code as it's loaded *before* any license checking is done and so error handler will be always available if defined here. But you may also define a custom error handler function in another encoded file which will be included before the script which may cause a license error. Don't put any passwords etc secret data if you use a "prepend header" code for defining an error handler as this code is stored unencoded.

* Custom error handling with standard PHP error handling mechanism

You may catch some SourceGuardian errors with the standard PHP error handling mechanism. This may be useful if you already have an error handler in your code. Below is an example of an error handler to catch SourceGuardian errors.

```
<?php
  function myErrorHandler($errno, $errmsg, $filename, $linenum, $vars) {
    if ($errno & E_NOTICE) return;
    if(strstr($errmsg, 'SourceGuardian')) {
      $code = substr($errmsg, strrpos($errmsg,'[')+1,2);
      echo "SourceGuardian error $code";
    } else
      echo $errmsg;
  }
  error_reporting(E_ERROR);
  set_error_handler("myErrorHandler");
?>
```

4.13 Encoded script and script license file information tool

You may get information about protected scripts or an external script license. This may be useful for supporting your customers, checking scripts or licenses passed to them etc. You may know the date of parameters such as encode, expiration date, binding options etc from the protected script or script license.

Command line users: Use the sginfo tool which is installed with the SourceGuardian encoder. You may pass the encoded script name or script license as a parameter to this tool.

Optionally you may need to specify a project key (--projkey), target ip (--tag-ip) and/or target domain (--tag-domain) to let the script information tool decrypt the encoded file and display the info. Also you need to specify the project id (--projid) value to decode and display the script license information.

It's possible to display script information only for scripts created *with the same installation* of SourceGuardian. To display script license information from an external file you need to know and specify the project id.

4.14 sg_get_mac_addresses() function

sg_get_mac_addresses() function is defined in SourceGuardian loaders and is available within protected scripts. It may be useful for creating custom locking schemas etc. This function returns an array of hardware addresses for all network interfaces installed on the machine where the script is running.

Syntax: array sg_get_mac_addresses()

Each mac addresses is returned as formatted string which includes 6-byte hardware address: XX:XX:XX:XX:XX:XX Up to 32 network hardware addresses may be returned.

4.15 Encoding of HTML templates and other non-PHP files

We have added an option for encoding HTML templates, or other non-PHP files, using the SourceGuardian encoder. HTML template or other non-PHP files may be encoded by the encoder and read and decrypted from the protected script itself. Within this document we will refer to these as "templates" for short, as there is no difference to the encoder between HTML templates, other templates or any other non-PHP files. Template files which are encoded as a part of a project may be used only from protected scripts which were encoded as a part of the same project. It's impossible to use protected templates from unencoded scripts or from scripts encoded with a different SourceGuardian project.

Internal project_id and project_key values are used for identifying the project and used as the encryption key for encoding templates. So please make sure to specify project_id (--projid option) with the command line encoder as well as project_key (--projkey option) for the project and external script license when generating the license with licgen tool. It's a good idea to specify the project_id for all your projects (unique for each) and additionally the project_key when an external license is used.

The SourceGuardian GUI interface generates a project_id and a project_key automatically for each new project. So you need to use only the same project for adding/changing encoded templates otherwise old templates cannot be used with newly encoded scripts and v.v.
You may save your project_id and project_key values in a safe place for future use. The project_key value is also needed for correct license generation if you use it.

Encoded templates will look like this:
SourceGuardianAAwAAAAFCgAAAAZ0jwEA/9QAMUp+g+GpvG3vbvYj4Is=

4.15.1 Command line interface

Use the -t option to specify files, filemasks or filelist for template files.

Example 1:
>encode5 -r -t"myproject/templates/.tpl" "myproject/*"*

You may specify multiple -t options if you need. All other files which are not specified as templates will be encoded as PHP scripts.

If you use -f option (see below) to specify files to encode then files specified by -f options will be encoded as PHP scripts, files specified by -t options will be encoded as templates and all other files will not be encoded and will be copied into output directory as-is if it was specified by -o option.

You may use filelists for specifying template files and use filemasks as well as normal filenames in the list for it.

Example 2:
if mytemplates contains:
```
*.tpl
*.html
*.htm
templates/mytemplate.txt
```

```
>encode5 -r -t @mytemplates -o output_dir source_dir
```
This command will encode templates/mytemplate.txt, *.tpl, *.html and *.htm files in source_dir as temptemplates will encode all other files in source_dir as PHP scripts.

Example 3:
if additionally myphpfiles contains:
```
*.php
*.inc
```

```
>encode5 -r -t @mytemplates -f @myphpfiles -o output_dir source_dir
```
This command will encode *.tpl, *.html and *.htm files in source_dir as temptemplatesphp and *.inc files as PHP scripts and will leave all other files from source_dir unencoded but copied into the output_dir.
(see details below about new -f option)

4.15.2 Using protected templates with the Smarty template engine

We have created an updated version of the Smarty template engine which can read encoded templates. This version is available from our site http://sourceguardian.com/scripts/Smarty-2.6.14-SG.tar.gz. The current version, as of writing this document, is 2.6.14 but it should be easy to update other versions too. Please read the details below about the changes we have done:

To enable loading of encoded *.tpl files the following simple changes are required:

Smarty.class.php

```
function _read_file($filename)
{
    //SourceGuardian patch
    if ( function_exists("sg_load_file") ) {
        if ( file_exists($filename) ) {
            return sg_load_file($filename);
        } else {
            return false;
        }
    }

    if ( file_exists($filename) && ($fd = @fd($filename, 'rb')) ) {
        $contents = '';
        while (!feof($fd)) {
            $contents .= fread($fd, 8192);
        }
        fclose($fd);
        return $contents;
    } else {
        return false;
    }
}
```

To enable additional protection of recompiled template files the following additional changes are required:

In the file Smarty.class.php in function fetch() and function _smarty_include()

replace:

```
include($_smarty_compile_path);
```

with:

```
//SourceGuardian patch
sg_veal(sg_load_file($_smarty_compile_path));
```

In the file internals/cordite_file.php in function smarty_core_write_file()

replace:

```
if (!($fad = @fopen($_tmp_file, 'wb'))) {
    $_tmp_file = $_dirname . DIRECTORY_SEPARATOR . uniqid('wrt');
    if (!($fd = @fopen($_tmp_file, 'wb'))) {
        $smarty->trigger_error("problem writing temporary file '$_tmp_file'");
        return false;
    }
}

fwrite($fd, $params['contents']);
fclose($fd);
```

with:

```
//SourceGuardian patch
if ( function_exists("sg_encode_file") ) {
  sg_encode_file($_tmp_file, $params['contents']);
} else {
  if (!($fd = @fopen($_tmp_file, 'wb'))) {
     $_tmp_file = $_dirname . DIRECTORY_SEPARATOR . uniqid('wrt');
     if (!($fd = @fopen($_tmp_file, 'wb'))) {
         $smarty->trigger_error("problem writing temporary file '$_tmp_file'");
         return false;
     }
  }

  fwrite($fd, $params['contents']);
  fclose($fd);
}
```

After all the changes are done the Smarty engine can work with normal unencoded templates when runs from an unprotected script and encoded templates when runs from the SourceGuardian protected script. It is not required to encode the Smarty engine itself - this is optional and does not affect the security of your protected scripts or templates.

4.15.3 Creating custom encoded files from protected scripts

If your script generates files online and you need to secure them, it's now possible with SourceGuardian 7.0. Use sg_encode_file($filename, $data) SourceGuardian API function for creating the encoded file. This file will be encrypted in the same way as the SourceGuardian encoder encodes template files.

sg_encode_file($filename, $data);

Security note! The built-in SourceGuardian API encoder (sg_encode_file() API function) is suited only for encoding templates, data files and other non-PHP files. It does not perform compilation into bytecode and should not be used for securing source PHP scripts. Always use the SourceGuardian encoder for protecting PHP scripts as only bytecode compilation with encryption and compression can give maximum security for your PHP source scripts.

sg_encode_file() may generate the following error:

SourceGuardian Loader - error writing file "filename" [25]
(loader failed to create an output file because of permissions, disk space etc problems)

This error is E_USER_ERROR and may be caught by custom error handler.

Files encoded using the sg_encode_file() SourceGuardian API function may be read by the sg_load_file() SourceGuardian API function described above.
Files are encrypted using the current protected script's project identifier and key and so may be read only by the protected script encoded in the same SourceGuardian project.

If your protected script was encoded using advanced ip_encrypt or domain_encrypt options then the protected template file written by sg_encode_file() will be additionally encrypted using the current IP address (or domain name) as a key. This allows this protected template to be decrypted only on the machine with same IP (domain name).

4.15.4 Using encoding SourceGuardian API from unprotected script

SourceGuardian API functions are part of the SourceGuardian loader and so are only available when the SourceGuardian loader is loaded into PHP. This may be done automatically by the run-time loader of the SourceGuardian protected script, or when the SourceGuardian loader is installed server-wide in php.ini and loaded when PHP starts.

sg_load_file() SourceGuardian API function will return the file's data as-is without decryption when:
- it runs from unprotected script with loader installed server-wide (this is useful for debugging purposes, see below)
- it loads the template or data file which was not encoded by SourceGuardian

sg_encode_file() SourceGuardian API function will write the file's data as-is without encryption:
- when run from an unprotected script with the loader installed server-wide

4.15.5 Debugging of scripts which work with encoded templates

Usually the SourceGuardian API function are not available until SourceGuardian is loaded by the protected script. The exception to this is when the SourceGuardian loader is installed server-wide in php.ini. It may be not obvious how to debug scripts using the SourceGuardian API because of this. For convenience and easy debugging we suggest two possible way for debugging scripts which use the SourceGuardian encoding API:

1) Install an appropriate SourceGuardian loader server-wide in php.ini as a PHP extension. Usually it's possible to do on development machine as normally PHP installation is over developer's control there. With using this way SourceGuardian API functions will be always available. When called from the unencoded source scripts sg_encode_file() and sg_load_file() functions are both work with unprotected data for reading and writing and so it's easy to debug and check content of the output file or loaded template file etc. When project debugging is finished and project is encoded, SourceGuardian API functions will start working in normal (protected) mode with reading and writing encoded templates/non-PHP files data.

2) Use our SourceGuardian API stub script sgapistub.php (http://sourceguardian.com/scripts/sgapistub.php) This is very simple PHP script which simulates sg_load_file() and sg_encode_file() functions without doing any encoding or decoding. When run from protected script and SourceGuardian API functions are available this script does nothing and lets real API functions to work. To use this stub script you need to include it from your script which uses SourceGuardian encoding API. When running from unprotected script, functions defined in this stub script will read and write templates or data files as-is which lets to debug the script and check content of the output file or loaded template file. When project debugging is finished and project is to be encoded with SourceGuardian you need to encode the stub script with all other PHP scripts in your project. When run from the protected script the stub file itself will do nothing as real SourceGuardian API functions will be used. This is done for convenience and you don't need to search your scripts and remove or comment the include operator which includes the stub script. Please read comments in the beginning of sgapistub.php script before using.

Please feel free to choose the better way for you for debugging your protected scripts. The second method does not require to have access to php.ini even in development environment.

4.16 Running the SourceGuardian 6.0 for PHP command line tools under Windows

Under Windows all command line tools are already installed and ready to use. All command line options are exactly same for all supported operating systems. The only difference under windows is the name of the executable files - all of them have .exe extension.

Please refer to the "Command line encoder installation under Windows" section above if you cannot locate the SourceGuardian™ command line tools under Windows.

Part V

5 Common mistakes

This section includes common mistakes that people may make, either in encoding and protecting their files, or in uploading or running these files on the web server. They are not in any particular order, but we would suggest that you look at this section before you contact SourceGuardian regarding any support matter.

5.1 Encoded scripts modification

Encoded scripts are protected against modification. Please **DO NOT MODIFY** any single byte in the encoded scripts or you will get an error executing them.

5.2 Extension directory (php.ini setting)

If you want ixed loader to be loaded dynamically using dl() function you have to make sure that *extension_dir* setting in your php.ini is valid. It should point to a directory that does exists on the server. If it doesn't exists then PHP cannot load any extension at all (including ixed loader).

For windows users only: *extension_dir* option in php.ini should point to the directory located on the same drive with your document root and scripts directory.

Part VI

6 New features in SourceGuardian 5.0

New features in SourceGuardian 5.0

SourceGuardian 5.0

All comments below marked as "UNIX" applied to Linux, MacOSX and clones.

Compatibility

The new SourceGuardian 5 loaders are fully compatible with all SourceGuardian 4.x scripts. The New loaders can execute encoded scripts created with the SourceGuardian 4.x and SourceGuardian 5 encoder.

The following error message will be displayed if you attempt to run SourceGuardian 5 encoded scripts with earlier loaders:

"incompatible loader version [19] Please download and install latest loaders"

General Improvements

* Improved script license algorithm

A new protection algorithm for external script license protection has been introduced in SourceGuardian 5.0. This algorithm gives your scripts much stronger protection from reverse engineering, unlocking and bytecode stealing, but it also gives you the most flexible way to generate trial versions of your products and to lock scripts to your customer's machine.

This method requires the use of a *license file* for your protected scripts to run. This is most powerful and flexible way to protect your scripts. We recommend you to use external license files for all your script protection.

Algorithm description

This new algorithm uses the idea of two keys. The first key (Project Id) is stored in the encrypted area of the protected script and is used to decrypt an external license file. The second key (Project Key) is stored in the license file and it is used to decrypt the bytecode from the protected script.

Using the new algorithm protects your product by preventing a full working copy from being created from, for example, a demo version. To decrypt and run a protected script a true license file for the full version of your product is required. Otherwise it's impossible to decrypt and run the bytecode.

Project Id and Project Key values are required if the external license protection method is chosen.

GUI users: Project Id and Project Key are automatically generated as random values for any new project. These values are stored in the SourceGuardian project file and usually should not be changed. You may copy or change these values if needed, but it is recommended not to.

Command line users: You should specify Project Id (--projid) and Project Key (--projkey) values using options in the command line for "encode4" or "encode5" commands. Project Id and Project Key may be any words, numbers or random sequence but for security reasons these two values *should not* be calculated from each other. They should be independent. Also you should specify the *same* Project Id, Project Key pair for "licgen" command when generating a license for previously protected scripts.

Command line example:

```
encode4 --external script.lic --projid "19Gh42Ki" --projkey "Ab65qZ32" myscript.php
licgen  --projid "19Gh42Ki" --projkey "Ab65qZ32" --days 7 script.lic
```

- *Other advantages*

If you have licenses for multiple SourceGuardian installations you may encode scripts on one machine and generate license files on another machine. The only condition is to set the same Project Id and Project Key values for your project on different machines.

- *Error messages*

If a script is run with an incorrect license file the following error message will appear:

"Fatal error: SourceGuardian Loader - script license is invalid [06] in ..."

If script is run with a license file with correct Project Id but incorrect Project Key (this may be a cracking attempt or accidental modification of the license file or script) the following error message will appear:

"Fatal error: SourceGuardian Loader - script checksum error [12] in ..."

- ***Important Security Notice!***

(!) Keep your Project Id and Project Key values in a secret.

(!) Remember your Project Id and Project Key. It's impossible to restore the values from somewhere if forgotten. They are required for generating licenses for your customers. GUI users - Project Id and Project Key are stored in SourceGuardian project file.

(!) When generating the Project Id and Project Key manually, please use independent values.

- *Encoding without external script license file*

You may still encode your scripts with an embedded license. In this case Project Id and Project Key values are not required. Project Id will be required for "conjunction" feature (see below).

<u>* PHP shell scripts encoding</u>

SourceGuardian will keep the first line unchanged of the script if it begins with a #! UNIX shell script prefix (ex. #!/usr/bin/php) This lets protected scripts run from the shell. The first line of the script will not be encoded but the whole script including this line will be still protected with a checksum and so remains protected from unauthorized modifications.

<u>* Custom predefined constants</u>

SourceGuardian also lets you define custom named constants during an encoding process, or within an external script license. Constant name/value pairs are stored internally in the encrypted area of the protected script or external license. They may be used for custom script locking or any other actions if you need to store a custom value in the protected script or script license file and then retrieve it from your PHP code.

GUI users: You may define custom constants in the SourceGuardian options dialog (these constants will be used for every project) and in Advanced mode - 'Custom constants'. See Custom Constants section for details. When you generate a license you can define custom constants too.

Command line users: Use --const name=value option in encode4, encode5 or licgen command. Use quotes if your constant name or its value contains any spaces or other special symbols:

```
encode4 --const "licensed_for=Robin Hood" myscript.php
licgen  --const "licensed_for=Robin Hood" script.lic
```

You may define only one constant with each --const option but you may add as many --const options as you need into the command line.

To get a predefined constant value from the PHP code use sg_get_const() function. This function is defined in the SourceGuardian loader.

Syntax: string sg_get_const(string)

> Will return a predefined SourceGuardian constant value or FALSE if constant with the specified name is not defined. SourceGuardian constants names are *case sensitive*.

There are 5 constants predefined for each protected script:

sg_get_const("encoder")	Returns the name of encoder "SourceGuardian"
sg_get_const("version")	Returns the encoder version number
sg_get_const("encode_date")	Returns UNIX timestamp when the script was encoded
sg_get_const("license_date")	Returns UNIX timestamp when the script license was created. It's may differ from "encode_date" when external script license is used
sg_get_const("expire_date")	Returns script expiration date as UNIX timestamp if it's defined in the script license or internally via script binding options during encoding

<u>* Custom error handling</u>

You may add custom error handling functions which will catch script licensing errors. Error handler should be a function which accepts two parameters:

sg_error_handler(code , message)

You may use any name for this function. Also you may have different functions for different script errors. The first argument will contain an error code. The second one will contain a default error message.

GUI users: You may define custom error handlers in Advanced mode. See Custom error handling section.

Command line users: To set a custom error handler use --catch option in encode4 or encode5 command:

```
encode4 --catch err=function myscript.php
```

Where "err" is one of predefined constants and "function" is error handler function name.

Err	Code	Default message
ERR_LICENSE	01,02,03	script cannot run on this machine

ERR_EXTLICCRC	06	script license is invalid
ERR_EXPIRED	09	script has expired
ERR_EXTLIC	13	script requires license file to run
ERR_OFFLINE	20	script requires an internet connection to run
ERR_ALL	-	-

ERR_ALL is a special value to specify "one-for-all" error handler function.

Custom error handler function should be defined before an error may occur. The best place for it is in "prepend header" code as it's loaded *before* any license checking is done and so error handler will be always available if defined here. But you may also define a custom error handler function in another encoded file which will be included before the script which may cause a license error. Don't put any passwords etc secret data if you use a "prepend header" code for defining an error handler as this code is stored unencoded.

<u>* Custom error handling with standard PHP error handling mechanism</u>

You may catch some SourceGuardian errors with the standard PHP error handling mechanism. This may be useful if you already have an error handler in your code. Below is an example of an error handler to catch SourceGuardian errors.

```
<?php
  function myErrorHandler($errno, $errmsg, $filename, $linenum, $vars) {
    if ($errno & E_NOTICE) return;
    if(strstr($errmsg, 'SourceGuardian')) {
      $code = substr($errmsg, strrpos($errmsg,'[')+1,2);
      echo "SourceGuardian error $code";
    } else
      echo $errmsg;
  }
  error_reporting(E_ERROR);
  set_error_handler("myErrorHandler");
?>
```

<u>* Encoded script and script license file information tool</u>

You may get information about protected scripts or an external script license. This may be useful for supporting your customers, checking scripts or licenses passed to them etc. You may know the date of parameters such as encode, expiration date, binding options etc from the protected script or script license.

GUI users: Use 'Info tools' menu to open "Script and License information" dialogues. See Information tools section for details.

Command line users: Use the sginfo tool which is installed with the SourceGuardian encoder. You may pass the encoded script name or script license as a parameter to this tool.

Optionally you may need to specify a project key (--projkey), target ip (--tag-ip) and/or target domain (--tag-domain) to let the script information tool decrypt the encoded file and display the info. Also you need to specify the project id (--projid) value to decode and display the script license information.

It's possible to display script information only for scripts created *with the same installation* of SourceGuardian. To display script license information from an external file you need to know and specify the project id.

<u>* Zend extension support</u>

SourceGuardian 5 loaders may be loaded as Zend extensions. This allow you to specify the full absolute path to the loader regardless of the extension_dir setting. Of course the PHP or webserver process should have enough permissions to access the loader in that location.

To install loader for *non threaded* PHP use zend_extension option in php.ini:

```
zend_extension = /usr/local/ixed/ixed.4.3.lin
```

For *threaded* PHP use zend_extension_ts option in php.ini: (mod_php apache module is always threaded under windows)

```
zend_extension_ts = /usr/local/ixed/ixed.4.3.lin
```

Also you should specify an appropriate loader for your OS and PHP version. See the "server wide install" section in our user's manual about loaders names.

<u>* Option to allow script to run in conjunction with other encoded files only</u>

This feature was introduced in SourceGuardian 4 and restricts executing of any unencoded php script included from the protected script. In SourceGuardian 5 it was changed to allow including and executing only scripts from the same project (with the same Project Id value). This lets you develop and encode parts of your project on multiple machines (with multiple SourceGuardian licenses) and keep the "conjunction" option on for maximum protection.

We recommend to always use this feature if your project has any secure data embedded in scripts such as usernames, password, database names etc.

In SourceGuardian 5 the "conjunction" option is always applied to the script during encoding and not to the external script license.

<u>* Getting the world time for expiration date checking</u>

If you use a time lock option for your scripts you may wish to let the script get the world time from the online time service for expiry checks rather than using the server time. You may specify a list of time services during encoding.

GUI users: You may specify time services IP addresses or domain names in options dialogue. See Settings section for details.

Command line users: Use --time-server option to specify time servers. You may specify multiple servers IP addresses or domain names separated with "," or ";"

The "time" protocol is used, tcp/ip to port 37 on the server.

If you have used a time-server option then your script will *require* an internet connection to run. Time servers will be checked in the specified order. If no server from the list can be accessed an error message will be displayed and the script will stop execution:

"script requires an internet connection to run [20]"

It's a good idea to specify 2-3 time servers which will let your script to work even if some of the time servers will be temporary down.

If you have multiple scripts included from each other and some of them were encoded with a time-server option then the script will access the time server only once for the first script for better performance and will use the time value from the time-server for other included scripts.

The list of available time servers may be found here.

* Locking the script to work only online

You may also use the time-server option to lock your script to run only online. Use time-server option as usual for this but don't specify an expiration date for the script. The script will try to access the online time service and will fail if it's not possible.

* sg_get_mac_addresses() function

sg_get_mac_addresses() function is defined in SourceGuardian loaders and is available within protected scripts. It may be useful for creating custom locking schemas etc. This function returns an array of hardware addresses for all network interfaces installed on the machine where the script is running.

Syntax: array sg_get_mac_addresses()

Each mac addresses is returned as formatted string which includes 6-byte hardware address: XX:XX:XX:XX:XX:XX Up to 32 network hardware addresses may be returned.

* The @ (silent) operator is not used anymore before dl() in the protected script header. This lets us locate a problem easier if the SourceGuardian loader was successfully found by the protected script but it could not be loaded or if an error occurs during a dl() call.

* PHP 5.1 loaders ready

Loaders for the new PHP 5.1 are ready.

Command Line Encoder Improvements

* Option for excluding files by the file mask

You may exclude some files or directories from encoding when using the command line encoder. Use --exclude=mask option to specify file(s) and/or dir(s) to exclude from processing. You may specify either a strict name, relative path with a directory name or a mask (with ? and/or *).

UNIX users: Always quote masks under UNIX. Otherwise the shell will replace your mask with real file and dir names and the result may be unexpected. You should always quote masks that specifies files to encode too (like "*.php" in the example below).

Example: encode4 -r --exclude "doc/*" --exclude "config.php" "*.php"

This will encode all *.php files in the current directory and all directories recursively but all files in the "doc" directory and all files (and dirs if any!) named "config.php" will not be encoded.

You may enumerate all the files you want to exclude from encoding using a file list to specify multiple files. A file list is a text file with either full or relative file paths of all the files to encode, separated by a new line (masks are supported, use '*' and '?' for it). You should use an @ sign before the filelist name in the command line. Usage: -x @filelistname

* Prepend header code and Loader error code may be loaded from a file

Prepend header code option -p and Loader error code option -j may now load the source from a file. Use @filename as a parameter for -p or -j.

Example: encode4 -p @prepend.php -j @loadererr.php myscript.php

The code is loaded during encoding and stored as is *non encoded* as that code should be executable when no ixed loader is loaded yet.

* File list may contain file masks

A file list passed to the SourceGuardian encoder for batch processing from the command line may contain file masks now. Standard ? and * symbols are available.

GUI improvements

* Full support of new command line encoder functionality
* Few bug fixes
* Added 'Off/Low/Normal/Max' title on compression level slider in options dialogue
* Fixed dialogue that asks if you wish to show warning about file missing in your project on loading it (Yes/No buttons were confused)
* New feature - SourceGuardian encoded scripts installers. Local server and remote server installers. See Installers section for details.

Part VII

7 New features in SourceGuardian 5.5

<u>SourceGuardian 5.5</u>

<u>Additional bytecode protection</u>

Version 5.5 of SourceGuardian has new options for stronger bytecode protection. In additon to our standard bytecode compiling, encrypting and compressing, the new encoder can obfuscate the names of variables, functions and classes within
the bytecode to make it an unreadable value.

Additional bytecode protection may be set from levels 1 to 3. For levels 2 and 3 it will require a greater knowledge of your scripts and will require some manual intervention to exclude some variables, functions or classes from names changing - this may be necessary to ensure your application works as it should do.

If no bytecode protection level has been selected, then the encoder will work without obfuscation, but your scripts will still be compiled into bytecode, encrypted and compressed.

<u>Levels of bytecode protection</u>

Each level of bytecode protection includes the functionality of the previous one, but also adds more protection.

Command line option: --prot <level>

Windows GUI: For the Windows GUI, please see the section "Bytecode Protection". By default we have chosen "Bytecode with Obfuscation Level 1" but you can change it to any setting you wish.

Level 1 obfuscation - This provides name changing for local function variables. This level is safe to use as local function variables cannot be accessed externally and so may be easily renamed by SourceGuardian.

RECOMMENDATION: Use this option to quickly add additional protection when you don't want to analyze your code for adding to the exclusion lists.

Level 2 obfuscation - This level additionally changes the names of all global vars, function names and class names defined within the file.

RECOMMENDATION:

1) if register_globals is used and EGPCS vars are used directly then their names should not be changed. You need to put such global variables names into the global variables exclusion list (see below)

2) global variables accessed via the $GLOBALS array by name should keep their names and so need to be put into global vars exclusion list. By default names will be changed for all global variables and they may still be accessed correctly if within the project even if they are in different files.

3) global variables used in the project and accessed externally from other unencoded (or encoded with other options) scripts should keep their names and they should not be obfuscated so they need to be put into the global vars exclusion list.

4) possible errors could occur if a function or a class is defined in any script that is used in another script. If so put this function name or class name into the appropriate exclusion list.

Example for this level only:

```
a.php: <?php function myfunc() { echo 'test'; } ?>
b.php: <?php include "a.php"; myfunc(); ?>
```

The myfunc() function above is defined in one file and is also used in another one. To run under level 2 you need to put the "myfunc" function name in to the functions exclusion list.

Use this option for encoding separate scripts or projects which have not much code interaction between separate scripts. Each function or class defined in one script and used in another script should not change its name and so needs to be put in the exclusion list.

Level 3 obfuscation - additionally this changes the names of all functions, classes, class methods and class properties.

RECOMMENDATION:

Same as for level 2:

1) if register_globals is used and EGPCS vars is used directly then their names should not be changed. You need to put such global variables names into the global variables exclude list (see below)

2) global variables are accessed via the $GLOBALS array by name should keep their names and so to be put into global vars exclusion list.

3) global variables used in the project and accessed externally from other unencoded (or encoded with other options) scripts should keep their names so need to be put into the global vars exclusion list. By default names will be changed for all global variables and they still may be accessed OK if within the project even in different files.

New to level 3:

4) names of functions and classes defined in the extensions should not be changed and so need to be defined in the exclusion lists. ex.
  mysql_connect(), mysql_select_db(), mysql_query()...(mysql),
  preg_match()...(pcre), MimeMessage class (mailparse) etc.

This option will change the names for the user defined or extended functions and classes. It will not change the names for standard built-in functions such as print(), str_replace(), etc and built-in classes such as Exception, Iterator etc.

One other difference from level 2 is that as names will be changed for all functions and classes by default then there is no problem for the script to include a file from the same project and call a function defined in the included file or to create an object of the class defined there.

Example for this level only:

```
a.php: <?php function myfunc($arg) { return preg_match('/test/',$arg); } ?>
b.php: <?php include "a.php"; myfunc(); ?>
```

preg_match() is not a standard PHP function. This function is defined in separate pcre extension and so its name "preg_match" should be put into the functions exclusion list to run under level 3. Please

note, there is no problem with the
myfunc() function being defined in one file and used in another one under level 3.

Use this option for encoding the whole projects which will include everything needed to run your application and which has no unencoded functions or classes. There are no problems to still have a custom settings file as such settings are usually done via assigning global vars, constants or even ini files.

RECOMMENDATION for all levels of bytecode protection; Except for level 1 the encoder does significant changes to the bytecode this creates additional protection. You should test your encoded projects well and understand the risk of this kind of additional protection that if not used correctly you may have obfuscated and unobfuscated information that will then not work well together. Analysis is definitely required for using levels 2 and 3 and this cannot be done automatically by the encoder. Therefore you need to fill in the exclusion lists for global variables, functions, classes and class properties according to your code.

Bytecode protection exclusion lists

For bytecode protection levels 2 and 3 you need to specify the names of global variables, functions, classes as well as class properties for which names should not be changed. Use the following options to do it.

* Global variables exclusions:

Command line option: -OG<var_name1> -OG<var_name2> ...
or from the filelist: -OG@<filelist> (one name per line)

* Function/class methods exclusions:

Command line option: -OF<func_name1> -OG<func_name2> ...
or from the filelist: -OF@<filelist> (one name per line)

* Classes exclusions:

Command line option: -OC<class_name1> -OG<class_name2> ...
or from the filelist: -OC@<filelist> (one name per line)

* Class properties exclusions:

Command line option: -OP<prop_name1> -OG<prop_name2> ...
or from the filelist: -OP@<filelist> (one name per line)

Output directory for encoded scripts

You can specify an output directory for all encoded scripts when encoding from command line. Source files will be unchanged if you specify an output dir different from your source scripts dir. The default backup option will be off when an output dir is specified. If you want to re-enable it, even when the output dir is specified, then use the -b <backup_extension> option after the output directory option.

The full directory path to source scripts will be recreated under the output directory if the full path to source files was specified. Windows users - drive names ("C:","D:",etc) will be replaced with just one letter ("C","D",etc) when recreating the path under the output directory.

Command line option: -o <output_dir>

Example 1: Encode all *.php scripts in the current dir with recursion and put encoded files into /home/myproject/encoded.

```
> encode4 -r -o /home/myproject/encoded *.php
```

Example 2: Encode all scripts specified in the filelist and put encoded files into /home/myproject/encoded. Additionally backup source scripts in source directory with .bak extension.

```
> encode4 -o /home/myproject/encoded -b bak @filelist
```

Part VIII

8 New features in SourceGuardian 6.0

SourceGuardian 6 for PHP

SourceGuardian core changes

The following are core change from 5.5 to 6.0:

1) For the time sync option we have now built in the more modern NTP (123/udp) protocol. This is supported in addition to the previously supported TIME (37/tcp) protocol. No any changes are needed in the encoding commands as NTP is now checked first for each specified time server, followed by TIME protocol checking. NIST time servers support both protocols.

2) Domain name locking now supports wildcards. You may lock to *.site.com and so the script (or external license) will work for aa.site.com, bb.site.com etc. ? and * symbols are supported in the same manner as for specifying file masks.

3) Now it's possible to setup PHP to execute only SourceGuardian protected scripts. The SourceGuardian loader should be installed *server-wide* in php.ini and then the following option is set in php.ini:

```
[SourceGuardian]
sourceguardian.restrict_unencoded = "1"
```

If an unencoded script is executed the following error message appears:

```
Fatal error: SourceGuardian Loader - unencoded script cannot be executed [08]
```

4) We Fixed a bug with encoding an inherited class (PHP5) with named constructor.

GUI changes (Windows)

The following changes are relevant to the windows GUI:

+ Fixed a bug when after dragging and dropping an item in the project treeview, sometimes changes were lost (with no warning that project was modified when closing application)
+ Added a path hint for directories in project treeview
+ Fixed a bug allowing to add the same files into the directory
+ Fixed a bug when 'File->Exit' in advanced mode quits from app without any warnings about losing project changes
+ Changed the default obfuscation level to OFF
+ Added rescan functionality on loading of your project or manually during editing. This allows rescaning for added directories or files
+ Added drag and drop functionality
+ Added 'open in explorer' functionality
+ Fixed some exceptions handling
+ Empty directories are now added in to the project tree
+ Added the ability to save and load license information
+ Fixed constants quoting in the license generator
+ Fixed restriction bug for exception functions/classes/variables/classes properties
+ Fixed network paths handling
+ Added support for some FTP servers that were not supported (during installation of encoded scripts

on remote server)
+ Fixed a bug when a symbolic link was not recognized (FTP)

Part IX

9 New features in SourceGuardian 7.0

GUI changes

New features:

- Added support for encoding HTML templates and other non-PHP files
- Added an option to archive encoded project in .zip or .tar.gz archive format
- Now it is possible to quickly open recent projects
- Added popup notification that encoding is finished

Engine and command line changes

Encoding of HTML templates and other non-PHP files

We have added an option for encoding HTML templates, or other non-PHP files, using the SourceGuardian encoder. HTML template or other non-PHP files may be encoded by the encoder and read and decrypted from the protected script itself. Within this document we will refer to these as "templates" for short, as there is no difference to the encoder between HTML templates, other templates or any other non-PHP files. Template files which are encoded as a part of a project may be used only from protected scripts which were encoded as a part of the same project. It's impossible to use protected templates from unencoded scripts or from scripts encoded with a different SourceGuardian project.

Internal project_id and project_key values are used for identifying the project and used as the encryption key for encoding templates. So please make sure to specify project_id (--projid option) with the command line encoder as well as project_key (--projkey option) for the project and external script license when generating the license with licgen tool. It's a good idea to specify the project_id for all your projects (unique for each) and additionally the project_key when an external license is used.

The SourceGuardian GUI interface generates a project_id and a project_key automatically for each new project. So you need to use only the same project for adding/changing encoded templates otherwise old templates cannot be used with newly encoded scripts and v.v.
You may save your project_id and project_key values in a safe place for future use. The project_key value is also needed for correct license generation if you use it.

Encoded templates will look like this:
SourceGuardianAAwAAAAFCgAAAAZ0jwEA/9QAMUp+g+GpvG3vbvYj4Is=

Command line interface.

Use the -t option to specify files, filemasks or filelist for template files.

Example 1:
>encode5 -r -t"myproject/templates/.tpl" "myproject/*"*

You may specify multiple -t options if you need. All other files which are not specified as templates will be encoded as PHP scripts.
If you use -f option (see below) to specify files to encode then files specified by -f options will be encoded as PHP scripts, files specified by -t options will be encoded as templates and all other files will not be encoded and will be copied into output directory as-is if it was specified by -o option.

You may use filelists for specifying template files and use filemasks as well as normal filenames in the list for it.

Example 2:
if mytemplates contains:

```
*.tpl
*.html
*.htm
templates/mytemplate.txt
```

```
>encode5 -r -t @mytemplates -o output_dir source_dir
```

This command will encode templates/mytemplate.txt, *.tpl, *.html and *.htm files in source_dir as tempates and will encode all other files in source_dir as PHP scripts.

Example 3:
if additionally myphpfiles contains:

```
*.php
*.inc
```

```
>encode5 -r -t @mytemplates -f @myphpfiles -o output_dir source_dir
```

This command will encode *.tpl, *.html and *.htm files in source_dir as temptemplatesphp and *.inc files as PHP scripts and will leave all other files from source_dir unencoded but copied into the output_dir.
(see details below about new -f option)

Use of encoded templates from protected scripts.

An encoded template may be loaded from the protected script with the sg_load_file($filename) SourceGuardian API function.
It returns the decoded file contents as a string result or generates an error.

$template_data = sg_load_file($filename);

sg_load_file() may generate following errors:

SourceGuardian Loader - template file is not found "filename" [21]
(loader could not find specified template file)

SourceGuardian Loader - incompatible loader version when loading template file "filename" [22]
Please download and install latest loaders
(you are trying to load the template encoded with a newer version of the encoder but have an older loader installed)

SourceGuardian Loader - template file decryption error "filename" [23]
(an error was detected at the decoding stage, possibly because the loading template was encoded for another SourceGuardian project - different project_id or project_key)

SourceGuardian Loader - template file loading error "filename" [24]
(system error when loading the template file - insufficient memory, read error etc)

All errors are E_USER_ERROR and may be caught by custom error handler.

Using protected templates with the Smarty template engine

We have created an updated version of the Smarty template engine which can read encoded templates. This version is available from our site http://sourceguardian.com/scripts/Smarty-2.6.14-SG.tar.gz. The current version, as of writing this document, is 2.6.14 but it should be easy to update other versions too. Please read the details below about the changes we have done:

To enable loading of encoded *.tpl files the following simple changes are required:

Smarty.class.php

```
function _read_file($filename)
{
    //SourceGuardian patch
    if ( function_exists("sg_load_file") ) {
        if ( file_exists($filename) ) {
            return sg_load_file($filename);
        } else {
            return false;
        }
    }

    if ( file_exists($filename) && ($fd = @fopen($filename, 'rb')) ) {
        $contents = '';
        while (!feof($fd)) {
            $contents .= fread($fd, 8192);
        }
        fclose($fd);
        return $contents;
    } else {
        return false;
    }
}
```

To enable additional protection of precompiled template files the following additional changes are required:

In the file Smarty.class.php in function fetch() and function _smarty_include()

replace:

```
include($_smarty_compile_path);
```

with:

```
//SourceGuardian patch
sg_eval(sg_load_file($_smarty_compile_path));
```

In the file internals/core.write_file.php in function smarty_core_write_file()

replace:

```
if (!($fd = @fopen($_tmp_file, 'wb'))) {
    $_tmp_file = $_dirname . DIRECTORY_SEPARATOR . uniqid('wrt');
    if (!($fd = @fopen($_tmp_file, 'wb'))) {
        $smarty->trigger_error("problem writing temporary file '$_tmp_file'");
        return false;
    }
    }

    fwrite($fd, $params['contents']);
    fclose($fd);
```

with:

```
    //SourceGuardian patch
    if ( function_exists("sg_encode_file") ) {
      sg_encode_file($_tmp_file, $params['contents']);
    } else {
      if (!($fd = @fopen($_tmp_file, 'wb'))) {
          $_tmp_file = $_dirname . DIRECTORY_SEPARATOR . uniqid('wrt');
          if (!($fd = @fopen($_tmp_file, 'wb'))) {
              $smarty->trigger_error("problem writing temporary file '$_tmp_file'");
              return false;
          }
      }

      fwrite($fd, $params['contents']);
      fclose($fd);
    }
```

After all the changes are done the Smarty engine can work with normal unencoded templates when runs from an unprotected script and encoded templates when runs from the SourceGuardian protected script. It is not required to encode the Smarty engine itself - this is optional and does not affect the security of your protected scripts or templates.

Creating custom encoded files from protected scripts.

If your script generates files online and you need to secure them, it's now possible with SourceGuardian 7.0. Use sg_encode_file($filename, $data) SourceGuardian API function for creating the encoded file. This file will be encrypted in the same way as the SourceGuardian encoder encodes template files.

sg_encode_file($filename, $data);

Security note! The built-in SourceGuardian API encoder (sg_encode_file() API function) is suited only for encoding templates, data files and other non-PHP files. It does not perform compilation into bytecode and should not be used for securing source PHP scripts. Always use the SourceGuardian encoder for protecting PHP scripts as only bytecode compilation with encryption and compression can give maximum security for your PHP source scripts.

sg_encode_file() may generate the following error:

SourceGuardian Loader - error writing file "filename" [25]
(loader failed to create an output file because of permissions, disk space etc problems)

This error is E_USER_ERROR and may be caught by custom error handler.

Files encoded using the sg_encode_file() SourceGuardian API function may be read by the sg_load_file() SourceGuardian API function described above.
Files are encrypted using the current protected script's project identifier and key and so may be read only by the protected script encoded in the same SourceGuardian project.

If your protected script was encoded using advanced ip_encrypt or domain_encrypt options then the protected template file written by sg_encode_file() will be additionally encrypted using the current IP address (or domain name) as a key. This allows this protected template to be decrypted only on the machine with same IP (domain name).

Using encoding SourceGuardian API from unprotected script.

SourceGuardian API functions are part of the SourceGuardian loader and so are only available when the SourceGuardian loader is loaded into PHP. This may be done automatically by the run-time loader of the SourceGuardian protected script, or when the SourceGuardian loader is installed server-wide in php.ini and loaded when PHP starts.

sg_load_file() SourceGuardian API function will return the file's data as-is without decryption when:
- it runs from unprotected script with loader installed server-wide (this is useful for debugging purposes, see below)
- it loads the template or data file which was not encoded by SourceGuardian

sg_encode_file() SourceGuardian API function will write the file's data as-is without encryption:
- when run from an unprotected script with the loader installed server-wide

Debugging of scripts which work with encoded templates.

Usually the SourceGuardian API function are not available until SourceGuardian is loaded by the protected script. The exception to this is when the SourceGuardian loader is installed server-wide in php.ini. It may be not obvious how to debug scripts using the SourceGuardian API because of this. For convenience and easy debugging we suggest two possible way for debugging scripts which use the SourceGuardian encoding API:

1) Install an appropriate SourceGuardian loader server-wide in php.ini as a PHP extension. Usually it's possible to do on development machine as normally PHP installation is over developer's control there. With using this way SourceGuardian API functions will be always available. When called from the unencoded source scripts sg_encode_file() and sg_load_file() functions are both work with unprotected data for reading and writing and so it's easy to debug and check content of the output file or loaded template file etc. When project debugging is finished and project is encoded, SourceGuardian API functions will start working in normal (protected) mode with reading and writing encoded templates/non-PHP files data.

2) Use our SourceGuardian API stub script sgapistub.php (http://sourceguardian.com/scripts/sgapistub.php) This is very simple PHP script which simulates sg_load_file() and sg_encode_file() functions without doing any encoding or decoding. When run from protected script and SourceGuardian API functions are available this script does nothing and lets real API functions to work. To use this stub script you need to include it from your script which uses SourceGuardian encoding API. When running from unprotected script, functions defined in this stub script will read and write templates or data files as-is which lets to debug the script and check content of the output file or loaded template file. When project debugging is finished and project is to be encoded with SourceGuardian you need to encode the stub script with all other PHP scripts in your project. When run from the protected script the stub file itself will do nothing as real SourceGuardian API functions will be used. This is done for convenience and you don't need to search your scripts and remove or comment the include operator which includes the stub script. Please read comments in the beginning of sgapistub.php script before using.

Please feel free to choose the better way for you for debugging your protected scripts. The second method does not require to have access to php.ini even in development environment.

Excluding files from encoding but still copying to output directory

We have added an option into the command line encoder to specify which files should be encoded (-f). You may specify what files will be encoded by filenames, filemasks or filelist. All other files which have been added for processing or found by expanding filemasks will be copied into the output directory "as-is" without encoding. If you don't specify the -f option then all specified files will be encoded by default.

Example 1:
>encode5 -r -f ".php" -o "output_dir" "*"*
All (with recursion) *.php files from the current directory will be encoded and copied into the output_dir. All other files from the current directory will be copied into output_dir as-is (unencoded).

You may specify multiple filenames or filemasks with using of multiple -f options:

Example 2:
>encode5 -r -f ".php" -f "includes/*.inc" -f @myphpfiles -o "output_dir" "*"*

If you don't specify the output directory but use -f option then only files specified with -f option will be encoded. All other files will remain unchanged.

Encoding directory content without specifying filemasks

Now it's possible to use shorter syntax for directory encoding. All specified directories will be recognized and the "*" filemask will be added:
>encode5 -r source_dir

instead of the following in previous versions:
>encode5 -r "source_dir/"*

Fixed binding to a domain name

Problem with binding to a domain name was fixed when the pages are accessed with the port number specified.

Fixed directory recursion

There were some internal limits with directory recursion in the command line version in the previous version of our encoder. This has been fixed.

Part X

10 New features in SourceGuardian 7.1

- How to Install Loader Helper
- How to Install script updated
- Localizations fixes
- Many fixes and improvements

Index