



Version 13 for macOS, Windows, Linux
User Manual

Introduction

An introduction to SourceGuardian 13 for macOS, Windows, Linux

by Team SourceGuardian

This SourceGuardian 13 for macOS, Windows, Linux 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 13 for macOS, Windows, Linux

© 2001-2022 SourceGuardian Ltd.

All rights reserved. No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of the publisher.

Products that are referred to in this document may be either trademarks and/or registered trademarks of the respective owners. The publisher and the author make no claim to these trademarks.

While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

Table of Contents

Foreword	0
Part I Introduction	2
1 About SourceGuardian for PHP	2
2 How to buy	2
3 Features	2
Part II GUI manual	7
1 Overview	7
2 Registration	7
3 Welcome screen	9
4 How to start	10
5 Project	10
Choosing files and destination	10
Locking options	13
Generating a license file	21
Advanced options	23
Encoding	27
6 Installing Loaders	28
7 Copying Loaders	30
8 Getting file information	31
9 Preferences and default settings	32
10 Viewing registration information	35
11 Getting help	35
12 Checking for update	38
Part III Command line encoder	40
1 Command line tools installation	40
Windows	40
Mac OS	40
Linux	40
Docker	42
First run	44
2 Running the command line encoder	44
Note for GUI users	44
Usage	44
Output directory for encoded scripts	47
Specifying which files to encode and which to copy	48
Excluding files from processing	48
Encoding entire directory contents	49
Locking options (full version only)	49
Advanced options	57
Custom predefined constants	61

Custom error handling	62
Other options	64
Encoding to standard output	68
Exit codes	68
3 Script license generator (full version)	69
Usage	70
4 File information tool (full version)	72
Usage	72
Part IV Running protected scripts	75
1 Installing loaders	75
2 Automatic loading	77
3 Loader filename structure	78
4 Zend extension support	79
5 Execute only SourceGuardian protected scripts	79
Part V Encoding of HTML templates and other non-PHP files	82
1 GUI	82
2 Command line interface	82
3 Using encoded non-PHP files	83
4 Using encoded templates with the Smarty template engine	84
5 Creating custom encoded files from protected scripts	85
6 Using encoding SourceGuardian API from unprotected script	86
7 Debugging of scripts which work with encoded templates	87
Part VI Common mistakes	89
1 Encoded scripts modification	89
2 Error messages during encoding	89
3 Error messages when running protected scripts	90
4 Extension directory (php.ini setting)	94
5 Getting "This protected script can run only in conjunction..." error	94
Part VII Advanced users	96
1 PHP shell scripts encoding	96
2 SourceGuardian loader API	96
3 Marking a file to be skipped during encoding	97
Part VIII License	99
1 SourceGuardian License	99
2 SourceGuardian Loaders License	104
3 Other Licenses for SourceGuardian for OSX	106
RegexKit Framework	106

Sparkle Framework	107
-------------------------	-----

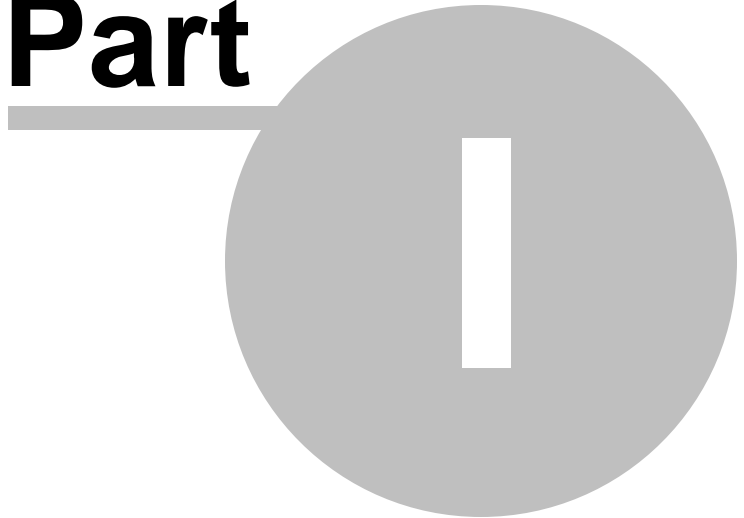
Part IX Changelog **109**

1 Version 13 / February 2022	109
2 Version 12 / February 2021	109
3 Version 11.4 / February 2020	110
4 Version 11.3 / April 2019	111
5 Version 11.2 / March 2018	114
6 Version 11.1 / April 2017	115
7 Version 11.0.6 / October 2016	118
8 Version 11 / June 2016	119
9 Version 10 / June 2014	120
10 Version 9.5 / July 2013	122
11 Version 9.0 / July 2012	123
12 Version 8.2 / April 2010	125
13 Version 8.1 / January 2010	126

Index **129**

SourceGuardian 13 for macOS, Windows, Linux

Part



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™ 13 for PHP, please visit our web site:

<http://www.sourceguardian.com/purchase/index.php>

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

1.3 Features

SourceGuardian™ 13 for PHP Features List

Protection method

The SourceGuardian™ 13 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.

Protected code

This sample PHP code:


```
<?php echo "Hello World!"; ?>
```

Becomes like this after encoding by SourceGuardian:

```
<?php
return sg_load('706A8B7A4CDFE9A0AAQAAAAAXAAAABHgAAACABAAAAAAAAD/
FEPsBWqk2zyokeO39F4Hw1eowZbkBfb9hEG
T7p9X+19m2X4Y+2KVUbfHkxhs9XAxgsPdQJvrot7Wi/6WA3bbiA/
SJekPYONSnvqMbfiYHWjDIUw1uCsB8vazpk3bATVdVKGrQyy
5D5Vx3YKbl/1abTdm5jCvP8SQAATAAAABbMouU4GsE2Y+bugvgoKydm8Oc9fcaKA4usVcazxOP0
TFyHd/QvNnro28hyaL+l7+8
blACHGBU5nL0a7QxU5s3zcl9PaDYmKqCicSwcn2jTH/4B3wkf4zFSFma7iZBuhJBg80HTXUcjr/4e52Gz
Ro43S7IKddTZ8sZiiGE
quFyUIBtvcpnFtIDKtU5n8Tpppq/gMVbDCHSgfig+KgQuLL/AAAAAA==');

```

Supported PHP versions

SourceGuardian™ 13 for PHP works with the following versions: PHP 4.3.x, 4.4.x, 5.0- 8.1 are fully supported.

Interface

SourceGuardian™ for Windows is a universal 32-bit GUI application which runs on 32-bit and 64-bit versions of Windows. Windows XP or later is required for encoding for PHP up to 7.1. Encoding for PHP 7.2+ require Windows 8 or newer with Microsoft Universal CRT redistributable package installed (installed automatically). A command line encoder is included. In addition, we also developed a powerful cross-platform GUI and command line encoders that runs under Macintosh and Linux.

Locking

To protect your scripts from unauthorised usage SourceGuardian™ 13 for PHP has added features that can optionally lock your scripts to run only from predefined IP addresses, domain names, LAN hardware addresses (MAC), machine ID. SourceGuardian™ 13 for PHP can also easily produce trial versions of your scripts by setting an expiry date for PHP scripts or by limiting the number of days that protected script will work. To protect against local date change for trial version of your protected scripts there is an option for time checking with atomic online time servers. For larger projects SourceGuardian™ 13 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 scripts may include a protected script from the unprotected script and this adds another level of protection.

Here is a sample list of features:

- locking to date with optional atomic online time servers checking
- locking to multiple domain names
- locking to multiple IP addresses
- locking to multiple LAN hardware (MAC) addresses, also works for CLI scripts
- locking to machine ID, also works for CLI scripts
- protecting of CLI scripts with remote verification URL

- improved locking to a specific domain/IP/machine ID with encryption. The domain name, IP or machine ID is used as a part of the key for encryption, so protected scripts may not be decrypted and run from another domain.
- 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.
- locking to an external license file produced by the built-in SourceGuardian™ 13 for PHP license generator. This is ideal for creating protected scripts to be deployed to different users and it even allows to assign different locking options to different users. The SourceGuardian™ 13 for PHP license generator tool can run from GUI or as a 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.
- locking protected scripts to work only online

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, then read and decrypted from the protected scripts code. Template files which are encoded as a part of the 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.

Other options

The following is not an exhaustive list, but covers some of the other options in version 13:

- PHP up to 8.1 are fully supported including new language features
- Improved code protection methods
- Encoding only files changed since last encoding
- A custom text may be added as is to the generated license file
- Built-in support for GUI versions
- option to check the date with online time servers (useful when locking to date is used)
- option to assign a custom PHP code to act as an error handler which will catch errors related to protected script loading or locking
- option to set a custom defined constant which may be read later from the protected PHP code
- automatic backup of source files
- multiple files processing: enumerated, file mask optionally with directory recursion or file list from the command line
- option to exclude some files from encoding process: enumerated or file masks
- option to specify an output directory for protected scripts
- encoding confirmation when run from the command line
- option to include PHP code to run before a protected script. This is best for including copyright information or for any other advanced needs
- option to replace the standard error handler when the appropriate loader is not found. PHP code can be included here
- Please see the [change log](#) for further details

Cross platform

Cross platform encoding. A script encoded under one operating system will run under any other supported operating systems. Currently we have the encoder for Windows, Linux and Macintosh. Protected scripts will work on Windows, Linux, OSX, FreeBSD, OpenBSD and other OS. Please find a list of all supported operating systems on [our web site](#). In the near future we will support more operating systems and can create a bespoke loader for your OS, please contact us support@sourceguardian.com if you are interested.

Thread Safety support

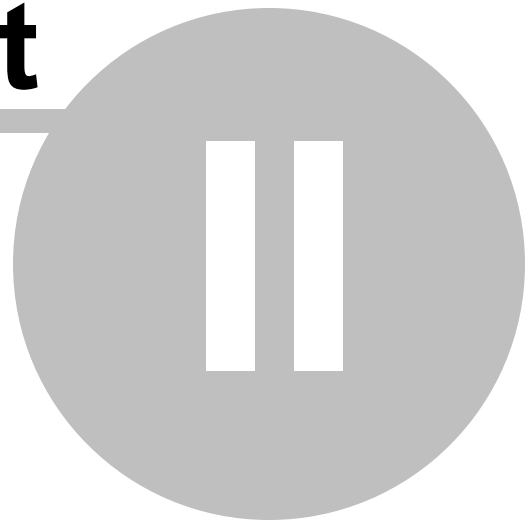
SourceGuardian™ 13 for PHP has a special versions of the loaders for Thread Safety PHP installations.

Evaluation

We provide a Free 14 days evaluation of SourceGuardian™ 13 for PHP. Locking options are not available in this free version.

SourceGuardian 13 for macOS, Windows, Linux

Part



2 GUI manual

2.1 Overview

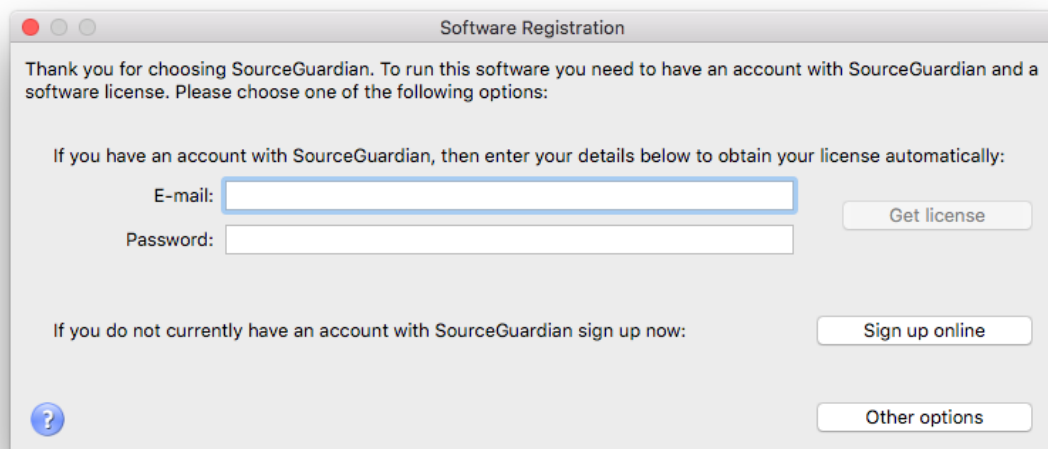


There are versions of SourceGuardian™ 13 for PHP that run on Windows, macOS and Linux. The GUI version is a user-friendly graphic interface to our command line encoder. It uses all the powerful features that command line encoder offers, while adding many additional useful enhancements to the encoding routine. User interface is easy to use and allows access to all powerful features of the encoder. The GUI has a multi-document interface which lets you work with multiple SourceGuardian projects at the same time.

Despite the screenshots in this document show SourceGuardian for Mac, GUI for Windows or Linux look exactly the same and include all the same options. Since version 11.3 of SourceGuardian we would like to keep a single version of the user manual which is easier to maintain and quicker to update than separate versions for different OS. Some functions of SourceGuardian are available with use of keyboard shortcuts, we will refer to them as Control/Command+Key below is the manual, which means using Control+Key for Windows/Linux and using Command+Key for Macs.

2.2 Registration

On your first run of SourceGuardian™ 13 for PHP you should see following screen:



This screen means that you need to obtain a license in order to run SourceGuardian™ . There are two ways to do this.

1. Obtaining a license from the application itself.

This is the fastest way to obtain a license, but to do it you need to have a connection to the Internet. If you don't have a connection to the Internet please use the second option ([see 2](#)) for obtaining the license using another machine connected to the Internet.

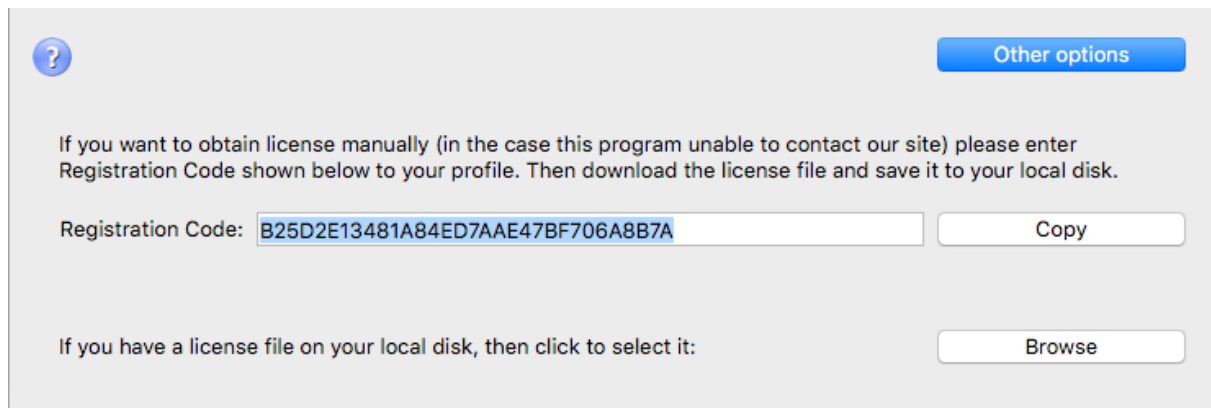
Please note, that some firewall or proxy software may prohibit SourceGuardian™ from connecting to the Internet, so you may have to enable the Internet access for SourceGuardian™ or obtain your license using another option ([see 2](#)). On how to enable the Internet access for a custom application with your firewall please consult your firewall documentation.

When you purchase a full version of SourceGuardian™ 13 for PHP, or request a demo version of it, you will receive an email with details on how to access your profile on our web site. This email contains a user name (which is your email address entered during registration) and a profile password. Please type them in the 'Email' and 'Password' fields and click on 'Get License' button. After the license has been successfully downloaded SourceGuardian™ 13 for PHP will run and you see a welcome screen. If anything has gone wrong with the installation, you will see an error message again. Make sure you have entered your email and password correctly, check your Internet settings and try again. If you cannot get a license please try another option ([see 2](#)).

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

If you are unable to obtain a license with the online registration method above, you can use this option to retrieve and download a license file. Go to the profile login page on the SourceGuardian.com website. Type your email and password.

Click "Other options" on Software Registration screen. You will see that some additional options become visible.



The screenshot shows a software registration window with a light gray background. In the top left corner, there is a blue circular icon with a white question mark. In the top right corner, there is a blue button labeled "Other options". Below this, a text block reads: "If you want to obtain license manually (in the case this program unable to contact our site) please enter Registration Code shown below to your profile. Then download the license file and save it to your local disk." Underneath this text, there is a label "Registration Code:" followed by a text input field containing the alphanumeric string "B25D2E13481A84ED7AAE47BF706A8B7A". To the right of the input field is a button labeled "Copy". At the bottom of the window, there is a text block that says "If you have a license file on your local disk, then click to select it:" followed by a button labeled "Browse".

Once you have entered your user profile on the web site, select the registration code from the SourceGuardian™ 13 for PHP application (by double clicking on it). Copy and paste it to the corresponding field in the "your profile" area on our web site ("Please enter your registration key here to generate license:").

When you have done the above, click on 'Submit' in the user profile on the web site and this will generate a license. To download it click on the 'Download' link in the 'Available licenses' section. Save this license, somewhere on your local disk. After that, in the SourceGuardian™ 13 for PHP application click on 'Browse' button, select the license file you have downloaded and saved. Software Registration window will be closed on successful license registration and you will see a welcome screen.

2.3 Welcome screen



This welcome screen appears when you start SourceGuardian™ 13 for PHP and there are no project files opened yet. If there are projects opened, you may open Welcome screen in Window/Open Welcome Screen menu. The following outlines features available on this screen:

Click on "Create a New Project" if you want to start a new empty project. Default options set in [File/Preferences](#) will be automatically applied to a new project.

Click on "Open Existing Project" if you already have a SourceGuardian™ 13 for PHP project and want to continue working with it, run encoding or generate a script license.

Using the top menu you can read built-in help, send support ticket to us, submit a feature suggestion and download latest loaders. See Help menu. Help is also always available by pressing Alt+F1 shortcut.

SourceGuardian™ 13 for PHP has a built-in support feature which you can use to send a question to our support team directly from the application. Click "Send Support Request" to do it from the Welcome screen. You may find a counter displaying a number of new responses received. Click on it or click on "View my requests" to open "My Support Requests" window where you can view responses and create new support tickets.

On the left hand side under "What's New?" you may find a feed from [our blog](#) and know the latest news

from us.

If you are in a trouble with starting using the encoder, please follow links under "Videos" and "Tutorials" on the Welcome Screen to learn how to use SourceGuardian.

2.4 How to start

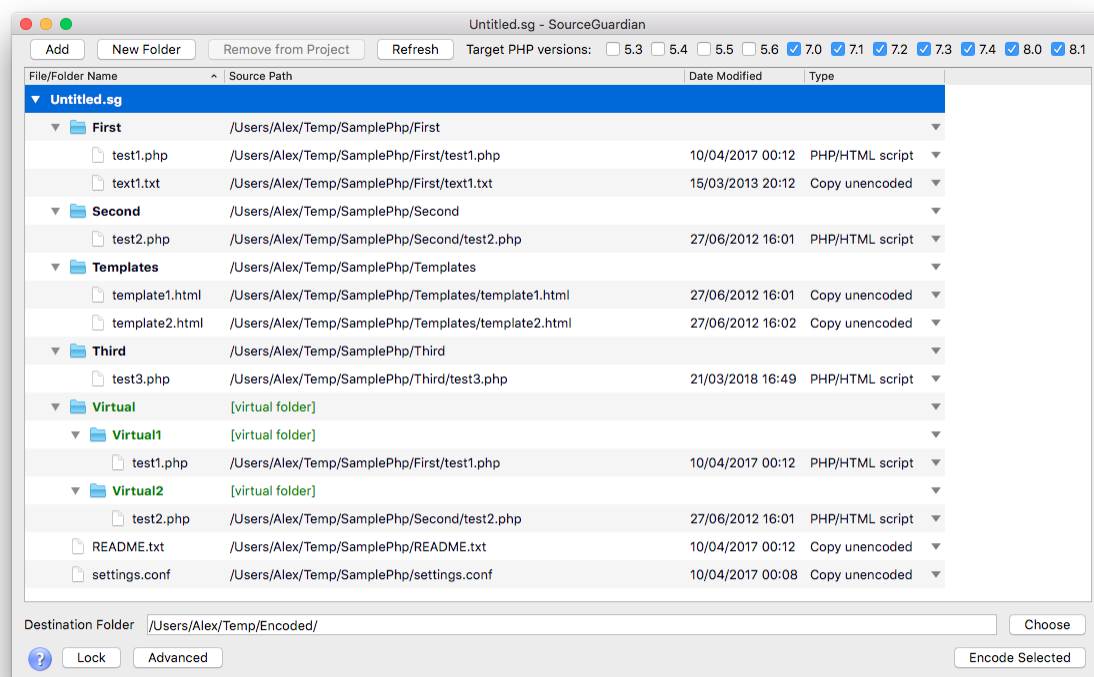
To start encoding of your PHP project please do 5 simple steps:

- 1) create a new SourceGuardian project
- 2) add files to the project tree
- 3) select versions of PHP your code is compatible with
- 4) select a destination folder
- 5) click Encode

Please refer to our [Project section](#) for further details.

2.5 Project

2.5.1 Choosing files and destination



The project window is the main window you will use when working with SourceGuardian encoder. It displays a SourceGuardian project which includes information about files to be encoded, encoding mode, target PHP versions, locking and advanced options. You can save the project to a file using File/

Save or File/Save As menu item. You may load a previously saved project using File/Open or choose from your recently used projects in File/Open Recent menu item.

Create a SourceGuardian project before you can encode files. This is a simple process during which you choose files to be encoded, select a destination folder, choose PHP versions, encoding mode and set encoding options. First two steps are required. All other steps are optional.

1. Choose files for encoding.

Clicking "Add" displays a dialog for selecting files or folders. You may either add separate files or add entire folders. Use "Remove from project" to delete files from the project that you do not want to encode. No real deletion happens, we only delete references to your files and folders from the project tree.

Choose encoding mode for files or folders. There are 4 modes available:

- PHP/HTML Script - the file will be encoded as PHP script or HTML. (If you encode HTML files, make sure your webserver is configured to run HTML files via PHP engine as all the files encoded in this mode are PHP scripts).
- Custom non-PHP - file encoded in this mode is encrypted and may be decrypted from your encoded PHP code using the `sg_load_file()` [SourceGuardian API](#) function. This mode is useful for encoding HTML templates, custom configuration files etc. Note, this mode differs from the "PHP script" encoding and files encoded as "Custom non-PHP" will not be automatically decoded by the SourceGuardian Loader. You need to use `sg_load_file()` API function to decrypt them from the protected code.
- Copy unencoded - the file will not be encoded, it will be copied as is to the destination folder. It's useful for files that you want to include to the protected project but which you do not want to encode at all, e.g. text, javascript, configuration, images, audio, video etc files.
- Skip - the file will not be encoded or copied to the destination folder. It's useful for files that you do not want to encode, but want to keep in the project tree.

To change encoding mode for a file or folder please choose it from the Type dropdown in the files list. Double click to select another encoding mode. You may select encoding mode for each file separately or set it for the entire folder. If you do it for a folder then selected mode will be assigned to all files and subfolders recursively.

When you add new files to the project, encoding modes are set automatically. This is done according to default settings in [File/Preferences](#). Feel free to change this according to file names and extensions you normally use in your projects.

You may build a new "tree" of your project instead of adding existing files and folders. Use "Create Folder" to create a new virtual folder within the project. You may add files and folders to the newly created folders as usual. Files added to the project will be copied to the destination folder and encoded there reproducing the folders structure of the project.

2. Choose a destination folder.

Choose a destination folder where encoded files will be copied to. Files marked as "Copy unencoded" will be copied to the destination folder as is without encoding. Click Choose button to select a destination folder. You may create a new folder by clicking "Create Folder" in the file dialog.

3. Optionally set target PHP versions.

SourceGuardian encoder produces different bytecodes for different PHP versions to provide maximum compatibility and full support of PHP language features. So you need to choose target PHP versions for

encoding. All versions of the internal bytecode for one source script will be packed into one protected script. Since version 8 of SourceGuardian you do not need to provide different versions of your protected code for different versions of PHP. Choose the target PHP versions by clicking checkboxes at the top of the project window. A new project will use default settings. You may change the default setting of the target PHP versions in [File/Preferences](#).

Your choice should be easy when you know what version of PHP is installed on the server where you plan to run your scripts. It is important to know which PHP versions your code is compatible with and do the correct choice of the target PHP versions. For example if your code uses array type hints, it will not be compatible with either PHP 4.x or PHP 5.0/5.1 and therefore you should tick PHP 5.2. Nothing bad will happen if you do a wrong choice. A built-in PHP compiler will try to compile your code for each of the selected versions of PHP and will display an error message in the case your code is not compatible with any of the selected versions of PHP. Note, the entire script will NOT be encoded in that case and you can find details about it in the encoding log.

Normally you will select the minimal supported version for your code and all the newer versions.

4. Optionally set locking options.

You may set various locking options which include IP address, domain name locks, hardware MAC address lock, expire date lock etc. Click Lock to set locking options. Please refer to [Locking options](#) section for further details.

5. Optionally set advanced options.

You may set advanced options which include special ASP and PHP tags support, custom header code, loader not found error handler etc. Click Advanced to set advanced options. Please refer to [Advanced options](#) section for further details.

6. Click Encode to start encoding.

If Encode button is not available then you have not added files for encoding or have not chosen the destination folder yet. Please do it to proceed with encoding. You may hold down the Control (Command) key on the keyboard before clicking Encode to encode only selected files and folders in the project tree. It is useful if you need to re-encode only some files without updating the entire project.

Files and folders highlighting in the project tree

Files and folders may be shown highlighted in the project tree. The legend is:
Folders - bold, virtual folders - green, files/folders changed or added since last encoding - blue.

2.5.2 Locking options

The 'Lock Options' window is a macOS-style dialog box with a title bar. It contains the following sections:

- Lock to a license file:** A checked checkbox. Below it is a text field 'Enter license file name:' with 'sample.lic' entered and a 'Generate License' button.
- Set expiration date:** A checked checkbox. To its right is a date picker showing '31.01.2022'.
- Script will expire in (days):** An unchecked checkbox.
- Script will only work with other encoded files:** An unchecked checkbox.
- Use remote verification (for CLI scripts):** An unchecked checkbox.
- Run from these IP addresses only:** A section with a '+' and '-' button. Below is a table with columns 'IP' and 'MASK'.
- Encrypt with IP as key (single IP only):** An unchecked checkbox.
- Ignore IP check for CLI:** An unchecked checkbox.
- Run from these domains only:** A section with a '+' and '-' button. Below is a table with column 'DOMAIN'.
- Encrypt with domain as key (single domain only):** An unchecked checkbox.
- Ignore domain check for CLI:** A checked checkbox.
- Online time servers for date check:** A section with a '+' and '-' button. Below is a table with column 'NTP SERVER'.
- Check date with online time servers:** An unchecked checkbox.
- Scripts will work only online:** An unchecked checkbox.
- Run from these machines only:** A section with a '+' and '-' button. Below is a table with column 'MACHINE ID'.
- Encrypt with machine ID (single machine ID only):** An unchecked checkbox.
- Custom defined constants:** A section with a '+' and '-' button. Below is a table with columns 'NAME' and 'VALUE'.
- Custom error handlers:** A section with a '+' and '-' button. Below is a table with columns 'ERROR TYPE' and 'HANDLER NAME'.

At the bottom left is a help icon (question mark in a circle). At the bottom right is a 'Close' button.

This window is used for setting locking options for encoded scripts. All locking options are stored encrypted within protected script. You may choose "Lock to external license file" option to put all locking settings into a special encoded license file which will be required for running your protected scripts. Please specify a [license file name](#) to use this option.

Set expiration date

Choose a date you wish your script to expire. The script will not run on and after the specified date and display the following error message: "Protected script has expired".

Script will expire in (days)

Select after how many days the script will expire starting from **today**. The script will not run on and after the specified date and display the following error message: "Protected script has expired".

Online time servers for date check

If you use an expiration date lock option for your scripts you may wish to let them check time with online time services rather than use local server time which may be potentially changed. You may specify a list of time servers. Old TIME and modern NTP protocols are supported. The list of time servers is automatically filled in for a new project using the default settings from [Preferences](#).

Please note, using this option will require the Internet access on the machine where your protected scripts run. Time servers are checked according to the list and if the first server is not replying then the second one is tried and so on. Using this option adds a delay to running your protected files because of accessing online time services via the Internet. If you use this option we suggest that you specify at least two time servers or more for reliability. If none of the specified time servers can be reached when your protected PHP code is running, it will fail with an error.

Time servers will be checked for protected scripts only if "Check date with online time servers" option is also selected.

Find further information about time locking, time servers and using the appropriate command line option [here](#).

Script will only work 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 of the same project and will NOT work if any of the included files or top files are substituted with an unencoded one or encoded as a part of another project or 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. Enabling this option makes it impossible to substitute c.php with their own code and do 'echo \$password' to know your password. Also enabling this option makes it impossible to create d.php which includes protected a.php and then does 'echo \$password'.

Note: this option must be enabled if you use [sourceguardian.restrict_unencoded](#) = "1" in php.ini

Run from these IP addresses only

Lock scripts to IP/mask. The encoder will lock the script to run only from the specified IP address(es). A specified IP address mask is applied to the real IP address before comparing. So you may use this option to lock the script to a subnet if a correct mask is specified. If a protected script is running from the IP address which is not allowed, the script terminates with the error message: "This script is not licensed to run on this machine". You may add as many IP address/mask pairs as you need. Click '+' button if you need to add another IP/Mask pair. Click '-' button if you want to delete the selected IP/Mask.

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

Please note, the loader will be able to check the domain name for a protected script ONLY in a web server environment. The web server should provide the PHP interpreter with one of the following environment variables SERVER_ADDR or LOCAL_ADDR. Apache web server does it by default for the CGI interface. It also provides these environment variables for the FastCGI interface, but it depends on the FastCGI server to provide these variables to the script. For the Nginx webserver you need to have an appropriate fastcgi_param line in the web server configuration file to have that variable passed to the PHP script. For other web servers please look at the configuration accordingly.

As a developer you may wish check if SERVER_ADDR or LOCAL_ADDR variable is available in the environment on the target server by printing \$_SERVER['SERVER_ADDR'] or \$_SERVER['LOCAL_ADDR'] values

Locking to IP addresses works only for scripts which will run on web servers. As there is no a definite IP address when the script runs from the shell (command line) you should NOT use locking to IP address for scripts which are intended to be run from shell e.g. for cron jobs.

Encrypt with IP

Lock and encrypt scripts to IP/mask. You may specify only one IP/mask. The encoder will lock the script to run only from the specified IP address. The encoder uses a specified IP address with an applied mask as a part of the key for encryption for providing maximum protection. SourceGuardian Loader will not be able to even decrypt the script that runs from the wrong IP address and display an error message: "Protected script checksum error". IP address mask 255.255.255.255 is used by default if not specified.

Locking to IP addresses works only for scripts which will run on web servers. As there is no a definite IP address when the script runs from the shell (command line) you should NOT use locking to IP address for scripts which are intended to run from shell e.g. for cron jobs.

Find further information about IP locking and using the appropriate command line option [here](#).

Run from these domains only

Lock scripts to a domain name. The encoder will lock the scripts to run only from the specified domain and all the sub domains. If an attempt is made to run the script on a non-authorized domain, the following error message is displayed: "This script is not licensed to run on this machine". You may add as many domain names as you need.

Please note, the loader will be able to check the domain name the protected script is running under ONLY in a web server environment. The web server should provide the PHP interpreter with one of the following environment variables; SERVER_NAME or HTTP_HOST. The Apache web server does it by default for a CGI interface. It also provides these environment variables for the FastCGI interface but it depends on the FastCGI server to provide these variables to the script. For the Nginx webserver you need to have an appropriate fastcgi_param line in the web server configuration file to have that variable passed to the PHP script. For other web servers please look at the configuration accordingly.

As a developer you may wish to check if the SERVER_NAME or HTTP_HOST variable is available in the environment on the target server by printing \$_SERVER['SERVER_NAME'] or \$_SERVER['HTTP_HOST'] values.

Use the name of the main domain in this option, not the name of any subdomain until you are sure you need to lock to a subdomain.

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.

You may use * and ? wildcards when specifying a domain name. Wildcards have their usual behavior similar to a file system.

Example 3: *.mydomain.com

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

Example 4: *mydomain.com (please note a change from the previous example)

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

Locking to domain names works only for scripts which will run on web servers. As there is no a definite domain name when the script runs from the shell (command line) you should NOT use locking to domain name for scripts which are intended to run from shell e.g. for cron jobs.

Encrypt with domain

Lock and encrypt scripts to one domain. You may specify only one domain. The encoder will lock the script to run only from the specified domain name. The encoder will use a specified domain name as a part of the key for encryption for providing maximum protection. The Loader will not be able to even decrypt the script that runs in the wrong domain name and will display an error message: "Protected script checksum error". You should not use wildcards for domain name when using this option. This option works ONLY for one strictly specified domain name.

Locking to domain names works only for scripts which will run on web servers. As there is no a definite domain name when the script runs from the shell (command line) you should NOT use locking to domain name for scripts which are intended to run from shell e.g. for cron jobs.

Find further information about domain locking and using the appropriate command line option [here](#).

Run from these MAC addresses only

Lock the script to LAN hardware (MAC) addresses (The "MAC" word used here has nothing about Apple Macintosh computers. MAC address is a hardware address of the local area networking LAN controller

available for all platforms). This address is usually unique for every 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 a colon (:). The encoder will lock a script to run only from the machine which has a networking adapter with a 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 the script on a machine that is missed a correct adapter, then the script fails with the error message: "This script is not licensed to run on this machine". You may specify multiple MAC addresses, if any one address matches then the script runs.

You may use 'ifconfig -all' command in Linux or OSX or 'ipconfig /all' in Windows to get a list of installed networking adapters and know MAC addresses.

Find further information about MAC address locking and using the appropriate command line option [here](#).

Run from these machines only

Bind the scripts to a machine ID. Machine ID is a unique identifier of the computer where your protected files are run. We use a special approach to know the machine ID and it differs for the platforms where SourceGuardian loader is installed. The encoder will lock scripts to run only from the machine which has the same machine ID as specified in the option. If there are more than one machine ID specified, then the protected code will run on any of these machines. If the script is run from another machine, the script fails with the error message: "This script is not licensed to run on this machine."

Use `sg_get_machine_id()` API method and get the machine ID of the computer where you run this method. It may be called directly or from the encoded PHP code (not locked with any options). Note, as this API method is binary compiled into the loader, an appropriate SourceGuardian loader must be installed before your code calls this method. We recommend that you create a mini project, encode it and run for obtaining the machine ID from the client's machine before locking to it.

Locking to a machine ID is an alternative to using MAC addresses locking for the scripts which are not running in a web server environment. E.g. running a script as cron task or a command line script.

Encrypt to machine ID

Bind and encrypt to a machine ID. The encoder will lock the script to run only on the machine with the specified machine ID. The encoder will also use the specified machine ID as a part of the key for encryption for enhanced security. The Loader will not be able to decrypt the script running from the invalid machine and will fail with the error message: "Script checksum error." This option works ONLY for one machine ID.

Find further information about machine ID locking and using the appropriate command line option [here](#).

Remote verification

This option lets you use a special approach for protecting and locking your PHP CLI scripts. Using of this option is preferred if your CLI script is a part of your encoded PHP WEB project. In that case this option may be used for locking CLI scripts to run only on the same machine where your WEB part of the project is installed and run. A CLI script encoded with `--remote-verification-URL` will not run on another machine and will fail with the error message: "The script is not licensed to run on this machine"

For this option to work you need to do two things:

1) Create a special encoded PHP script which is accessible via HTTP protocol, it will be used to validate the machine and return the verification ID to the CLI script. The only directive you need to put into it is:

```
<?php
    echo sg_get_verification_id();
?>
```

Note 1: if you use any frameworks, you may need to use another mechanism for sending a string to the output stream instead of *echo*

Note 2: as `sg_get_verification_id()` API method is binary compiled into the loader, make sure you have encoded the verification script.

Normally, you will add this to your WEB part of the project and encode it with SourceGuardian along with the other web PHP files. You are free to use any name for this validation script. *Example:* `verification_id.php`

2) When encoding your CLI script, specify the full URL to your web verification script in this option and make sure you are encoding both your CLI script and the verification script as parts of the same GUI project, or, if you prefer to use separate encoding projects, use the same project ID and Key for the web part of your code and the verification script.

Example: `http://mydomain.com/verification_id.php`

Note 1: You may use standard locking to IP or domain for your web verification script (as usual for this to work your web server must send the information about current IP and domain to PHP via environment). Anyway, you "lock" to the domain if use the domain name in the verification URL or lock to IP if you use IP in the URL - choose the way which suits your project and the deployment process.

Note 2: you should not worry about the delays HTTP adds, they are minimal as your CLI code and the main WEB code are both running on the same machine. However, it's recommended to check the verification URL is accessible from CLI, e.g. using 'curl' or 'wget'. If the domain name can't be recognised from CLI, you may add the domain name to hosts file or switch to using IP instead of the domain name.

However, if your CLI PHP script works on its own and is not a part your your web based project, then you still may use locking to a machine ID described above as well as good old locking to MAC addresses.

Find further information about remote verification and using the appropriate command line option [here](#).

Script will only work online

If you do not use time locking option you are still able to lock the scripts to work only online. Select this option if you need your protected scripts to work only online (when the Internet connection is available). SourceGuardian Loader will check if the script it is running online by trying to access one of the time servers specified in the list. The list should not be empty! As mentioned above, this may add delays to executing of your protected files.

Custom defined constants

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

You may define multiple name/value pairs for your custom constants.

To get a predefined constant value from the **encoded script** use `sg_get_const()` [SourceGuardian API](#) function. This function is defined in the SourceGuardian loader and returns a predefined SourceGuardian constant value or NULL if constant with the specified name is not defined. SourceGuardian constants names are **case sensitive**.

Syntax: `string sg_get_const(string)`

There are 5 constants predefined for all protected scripts:

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

Custom error handlers

You may add custom error handling functions which will catch script licensing errors. An error handler is 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 is an error code. The second one contains a default error message.

The custom error handler function must be defined before a script licensing error may occur. The best place to define it is to use "custom header code" (see [Advanced options](#)) This header code is loaded before any license checking is done and so error handler will be always available if defined there. But you may also define a custom error handler function in another encoded file which is included before the script which may cause a license error. Don't put any passwords etc secret data if you use a "custom header code" for defining the error handler as this code is stored unencoded.

There are following error handler conditions defined:

Err	Code	Default message
Restricted to run	01,02,03,04,05	This script is not licensed to run on this machine. (code indicates a reason: 01-IP, 02-domain, 03-MAC address, 04-machine ID, 05-remote verification URL locking)
License broken	06	A license file which is required to run this protected script is invalid...
License	09	This script has expired...

expired		
License not found	13	This protected script requires ... license file in order to run
Running offline	20	This protected script requires the Internet connection in order to run
All errors	-	-

"All errors" is a special value to specify one error handler function for all SourceGuardian error codes.

The custom error handler function should be defined before an error may occur. The best place for it is "[custom header](#)" code as it's loaded before any license checking is done and so the error handler will be always available if defined there. But you may also define a custom error handler function in another encoded file which is run before the script that may cause a license error. Don't put any passwords or secret data if you use a "custom header" code for defining an error handler as this code is stored unencoded.

Find further information about custom error handling and using the appropriate command line option [here](#).

External license file name

If you use the "Lock to external license file" option then you need to specify a license file name. Scripts encoded with this option will require a license file in order to run. If the name of the license file does not include any path then protected scripts will search for the license file in the protected script's folder, then its parent folder and so on. So you may have one license file for the entire project if the license file is located in the project's root. Alternatively, an absolute full path may be specified for the license. If the protected script cannot find the specified license file, it fails with the error message: "script requires ... file to run".

The license file may be [generated later](#) using GUI or the [command line license generator](#) "licgen".

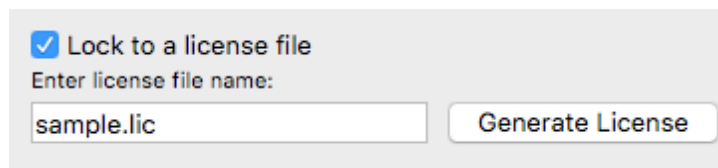
Locking your scripts to an external license file gives additional protection to your scripts. Using locking to a license file is the best if you need to deploy one script or the entire project to different users, but need to use different restriction options for each of them. Find more about how it works [here](#).

Find further information about custom error handling and using the appropriate command line option [here](#).

Generating a license file

To generate a license file click "Generate License" button. Please refer to [Generating a license file](#) section for further details.

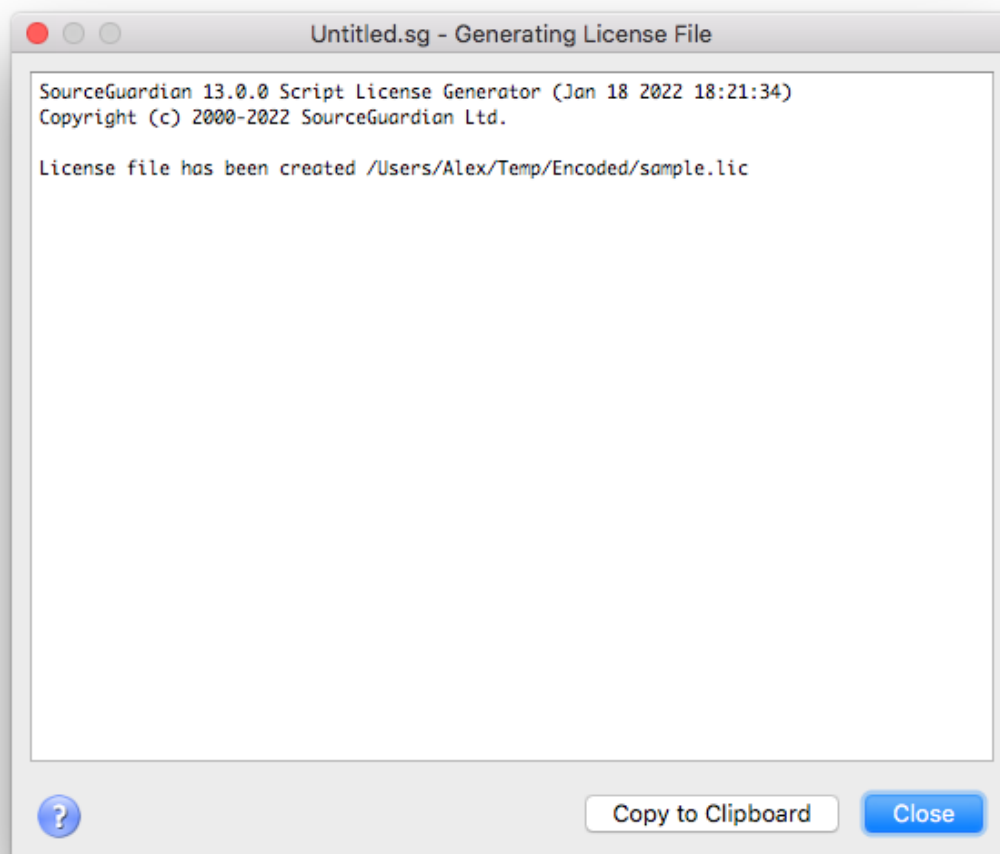
2.5.2.1 Generating a license file



A dialog box with a title bar. It contains a checked checkbox labeled "Lock to a license file". Below it is the text "Enter license file name:". There is a text input field containing "sample.lic" and a button labeled "Generate License".

When you select an option to lock to external license file, all the locking options are stored within it and that file is required for running protected script. In order to generate a license file please open locking options in the project window and click "Generate" on the top next to the External license file name field. You should already have "Lock to external license file" option selected and a license file name filled in as you used this locking mode for scripts.

The license generator takes all the locking options from the Lock screen and puts them into the license. It's OK to not have any locking options specified. In that case the license file will allow to run your protected files without setting any additional restrictions on them.



A new popup will show the license generation log. It will take a second and you see a message saying that the license file has been successfully created. The generated license file may be deployed along with your protected scripts or sent separately to your customers.

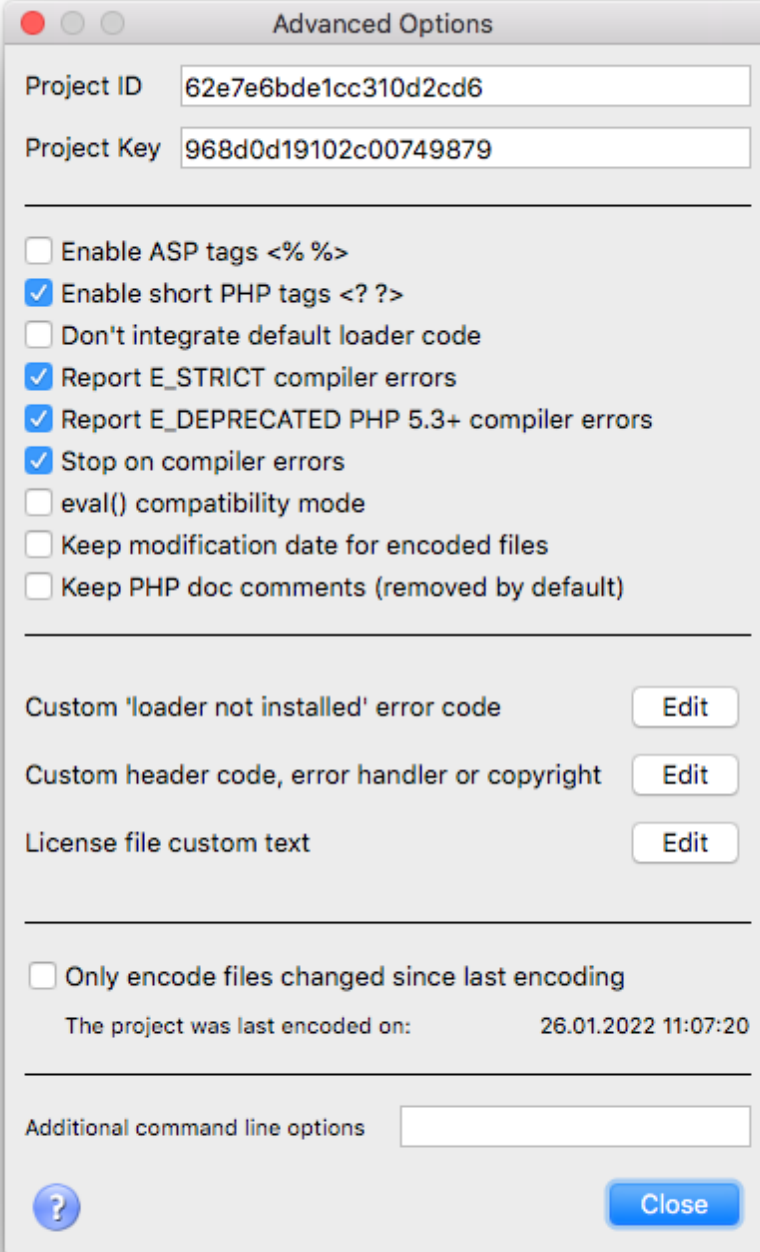
Normally, you will use a short name for your license file without a folder. In that case, when running protected files the license file will be searched in the encoded script's folder, then if not found, in the parent folder and so on. If you lock your project to a license file and use a short name for the license, you or your customers will need to install the generated license file to the root of your encoded project. In that case all the locked protected files will be able to find and load the license.

A full path may be specified for the license file. In that case when your protected files run, the license file will be checked by the full absolute path. When you click "Generate License" to create it, the full path is ignored and the license file will be created in the selected folder. However, the full path is stored along with the required license file name.

You may change locking options and generate different license files if you need.

Also you may automate license generation using a command line license generator tool. Using the command line license generator provides a method for licenses to be dynamically created and this may be useful when you are selling your scripts online or to automate your web site's backend etc. Please read about using the command line license generator in the "[Using external script license generator](#)" section in this manual.

2.5.3 Advanced options



Advanced Options

Project ID

Project Key

☐ Enable ASP tags <% %>

☒ Enable short PHP tags <? ?>

☐ Don't integrate default loader code

☒ Report E_STRICT compiler errors

☒ Report E_DEPRECATED PHP 5.3+ compiler errors

☒ Stop on compiler errors

☐ eval() compatibility mode

☐ Keep modification date for encoded files

☐ Keep PHP doc comments (removed by default)

Custom 'loader not installed' error code

Custom header code, error handler or copyright

License file custom text

☐ Only encode files changed since last encoding

The project was last encoded on: 26.01.2022 11:07:20

Additional command line options

Project ID

This field lets you assign ID to your project that is used to identify what license it should accept. Specify

the same Project ID in the license generator when you generate a license file for this project (if you use a command line license generator tool). This option is useful when you want to deploy several products that use external license locking so that each license works only with a corresponding Project ID.

Project Key

This field is used in conjunction with Project ID and required if you plan to use external license locking. Introduced in version 5.0 a new algorithm uses an idea of two keys. The first key (Project Id) is stored within the encrypted area of protected scripts and used to decrypt an external license file. The second key (Project Key) is stored within the license file and used to decrypt the bytecode from the protected script.

This algorithm protects your product against creating a full working copy from the demo version by some people who may be interested in this. In order to decrypt and run the protected script a valid 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 external license protection method is selected.

Project ID and Project Key values are randomly generated for a new project. Usually you do not need to change them. If you need to use the same Project ID or Project Key value for any reasons (e.g. when creating different projects which share the same license file) you may change the values in Advanced options manually.

Enable ASP tags

Enables use and recognition of ASP-like <% %> tags for indicating the PHP code in addition to standard <?php ?> tags.

Enable short PHP tags

Enables use and recognition of short PHP tag <? for indicating the PHP code, otherwise only standard <?php and ?> tags are recognized.

Don't integrate default loader code

You may use this option if you don't want to include the default starter code into protected scripts. Scripts encoded using this option will not be able to automatically find and load an appropriate SourceGuardian loader and you have to install the ixed loader manually to run this script. See this [section](#) about the manual ixed installation. If you already have the SourceGuardian loader installed server-wide in php.ini then this option may be useful.

Since PHP 5.2.5 dynamic extensions including SourceGuardian loaders must be installed to PHP's extension_dir folder specified in the php.ini configuration file and an appropriate extension=ixed.XX.YYY directive must be added to the php.ini in order to install the loader.

Note: if you select this option then "Loader not found error code" option has no effect (as the code is placed inside the default starter code).

Report E_STRICT compiler errors

E_STRICT "Strict Standards" warnings were introduced in PHP 5. This option instructs the encoder to

warn of such messages during encoding. This option is ignored when encoding scripts in PHP 4.x mode. Usually E_STRICT warnings may be ignored but it may be a good idea to let the encoder display such warning messages and review the code. The encoder will stop if "Stop on compiler errors" option is also checked.

Note: the encoder can catch only compiler-related E_STRICT warnings. Run-time E_STRICT messages will be displayed when the protected script runs as usual according to the error_displaying option in the php.ini.

Report E_DEPRECATED PHP 5.3+ compiler errors

E_DEPRECATED warnings were introduced in PHP 5.3. This option instructs the encoder to warn of such messages during encoding. This option is ignored when encoding scripts for PHP older than 5.3. Usually E_DEPRECATED warnings may be ignored but it may be a good idea to let the encoder display such warning messages and review the code. The encoder will stop if "Stop on compiler errors" option is also checked.

Note: the encoder can catch only compiler-related E_DEPRECATED warnings. Run-time E_DEPRECATED messages will be displayed when the protected script runs as usual according to the error_displaying option in the php.ini.

Stop on compiler errors

This option instructs the encoder to stop encoding at first critical error or E_STRICT/E_DEPRECATED warning if these options are selected. This may be useful if you have many files in the project and there is a risk of missing errors and leaving some files unencoded because of it.

Note: Even if this option is off you will be able to find all error messages in the encoding log.

eval() compatibility mode

Enables eval() compatibility for encoded scripts. Normally encoded scripts cannot be run with the PHP eval() function. Additional CRC check restricts it as the protected code is passed as a string and source file is unknown. This improves security for encoded scripts that run in a standard way. However, some PHP template engines or custom code requires loading encoded PHP scripts as a string and then passing it to the eval(). In order to enable running protected scripts with eval() you may use this option and encode those files in the 'eval compatibility' mode.

Keep modification date for encoded files

This option instructs the encoder to keep the modification date for encoded files the same as modification date of source files. This may help in deploying only updated files and in some other cases of custom deployment of encoded files. The modification date for encoded files is set to the current date by default if this option is not used.

Keep doc comments

PHP doc comments are removed by default. Use this option to keep them e.g. if the framework you use

require doc comments. We also added an appropriate CLI option `-keep-doc-comments`.

Custom header code

This options lets you add a custom header at the top of every encoded file. You may put any code to be executed BEFORE the protected scripts code. This code WILL NOT BE ENCODED (although it is still protected with CRC against changes). This may be either HTML text or PHP code. For PHP code - you should enclose your code with `<?php ?>` tags. This option is usually used for including copyrights into protected scripts but also it's used for including [custom error handler](#) functions.

All user {constants} that are defined in [locking options](#) will be replaced in the prepend code. Also some standard SourceGuardian constants may be used:

{SG_DATE} - current date i.e. date of encoding
{SG_LICENSEE} - SourceGuardian license owner from the SourceGuardian license file (your name)

Constant names are **case sensitive**. It works in the same way also for `licgen` and `do` replacements for custom text if it is used. [See details](#)

Loader not installed error code

It is possible to change the default action when an appropriate SourceGuardian loader is not installed and could not be found or used for automatic dynamic loading. The default handler included into protected script starter's code (which prepends each protected script by default) will display an error message "This script is protected by SourceGuardian™ and requires file ... " and then the script stops executing. This option allows you to change the default error action. You may use any HTML text or PHP code and it will be displayed or executed as a replacement to the default SourceGuardian™ loader error. This code WILL NOT BE ENCODED (although it is still protected with CRC against changes). This may be either HTML text or PHP code. For PHP code - you should enclose your code with `<?php ?>` tags.

License file custom text

If you use locking to an external license file you may add custom text that will be embedded as is into generated license files. The text is protected with a checksum against modification. You may include any text as user information, license description etc into license files. One text will be used for all license files unless you change it.

All user {constants} that are defined in [locking options](#) will be replaced in the text. Also some standard SourceGuardian constants may be used:

{SG_DATE} - current date i.e. date of encoding
{SG_LICENSEE} - SourceGuardian license owner from the SourceGuardian license file (your name)
{SG_EXPIRY_DATE} - expiry date of the custom license you generate. If the expiry date is not set, a word "never" will be placed into the text

It works in the same way also for the custom header in protected scripts. [See details](#)

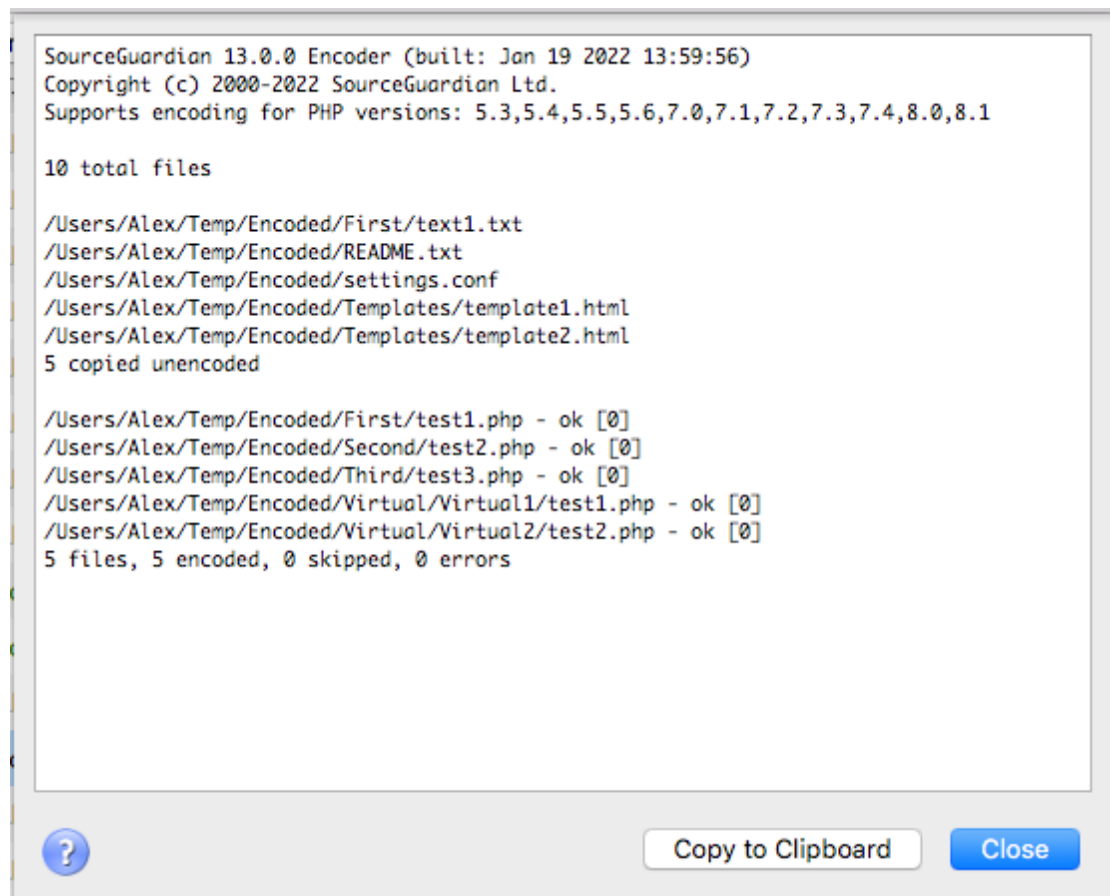
Only encode files changed since last encoding

This option lets you encode only files that have been changed since when you encoded the project for the last time. The encoder saves the date and time in the project file when the project was encoded last time. When this option is used the encoder checks your source files added to the project to find which files have been modified. This happens automatically when you open the project and you may turn this off/on in [Preferences](#).

Additional command line options

This option lets you pass additional options to the command line encoder when it's being called for encoding the project. Normally, you do not need to add any options as all of the options are available on the 'Lock' and 'Advanced' windows in GUI. **Use this on your own risk. Specifying unexpected options to the command line encoder may cause unexpected results or even files loss.** If there are any issues with encoding of some code or issues with running encoded code, our support team may suggest that you pass some special options to the encoder using this line for debugging purposes.

2.5.4 Encoding



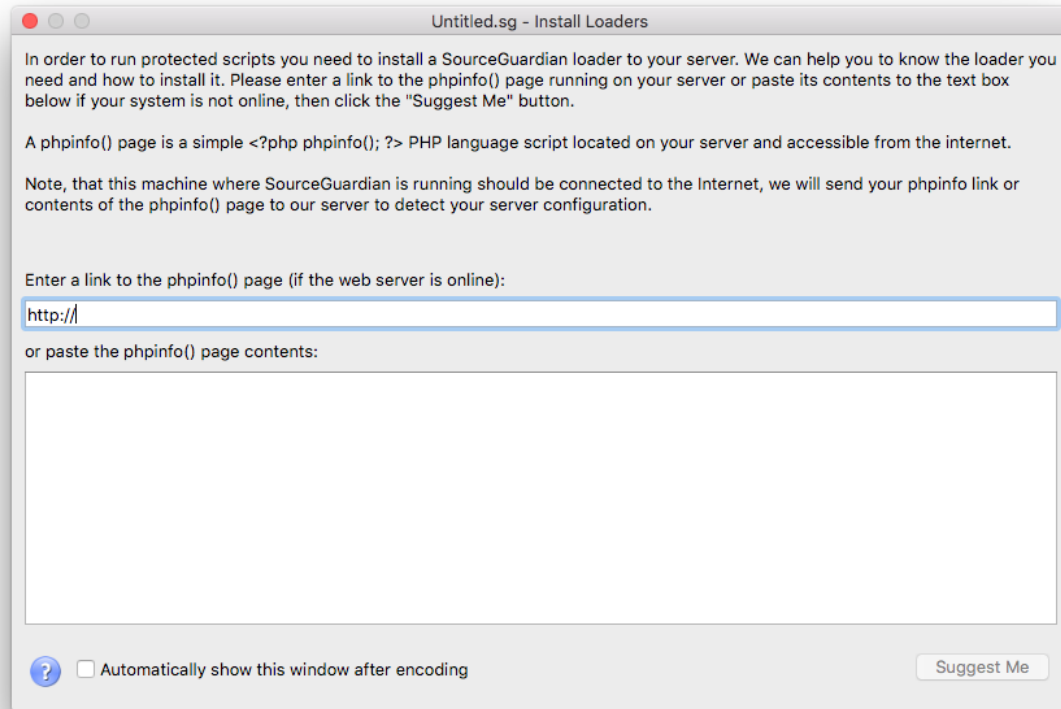
After you have added files to your project, selected a destination folder and optionally set locking and advanced options you may start encoding of your scripts. This is simple - just click on "Encode" in [project window](#). A popup window will display the log of encoding. At the end of the log you can find the number of processed files and files which could not be processed because of errors.

You may see the following error messages next to file names:

Error message	Description
ok	The script, template or other non-PHP file was encoded without problems.
file not found	File for encoding was not found (this error should not appear in the GUI).
PHP compiler error	You have an error in your script and the encoder could not compile it into bytecode. This message is specific to the PHP version which is also displayed as well as the line number. Check if your script is free from errors and that <u>it is compatible with the version of PHP you are encoding for</u> . E.g. using of some new language features are only possible with recent versions of PHP. Encoding such scripts for older versions may result in the error message.
could not write file	The destination folder that you selected is not writable.
file is already processed by SourceGuardian	File is already encoded with SourceGuardian and could not be encoded once again (this error should not appear in the GUI).
empty file, skipped	Empty files could not be encoded and not processed. If you need to have this empty file in the destination folder, change encoding mode to "Copy unencoded" for it.
not regular file, skipped	The file you have selected for encoding is not a regular file (for example it's a device driver file etc)
copied	This is normal. You have selected "Copy unencoded" for this file and it was copied to destination folder.
internal encoder error	Please let us know about this error. Send us a support ticket including the file that caused this error.
unknown error	Please let us know about this error. Send us a support ticket including the file that caused this error.

You may stop encoding by clicking "Stop". Once encoding is finished click "Close" to close the encoding log window and return to your project.

2.6 Installing Loaders



We have created this "Install Loaders" wizard in order to help you know the loader you need and how to install it to your target system where protected PHP files will run. Please read the text about SourceGuardian loaders on this screen. This wizard checks your remote system and knows how to install the loader to it. You need to create a simple phpinfo page on your remote machine. The phpinfo page is a code like this (without quotes) "<?php phpinfo(); ?>" which is located on your remote system and available from the Internet. Please enter a HTTP link to access the phpinfo page on your remote machine and click "Suggest Me" button to get instructions of how to download and install an appropriate loader to your remote machine.

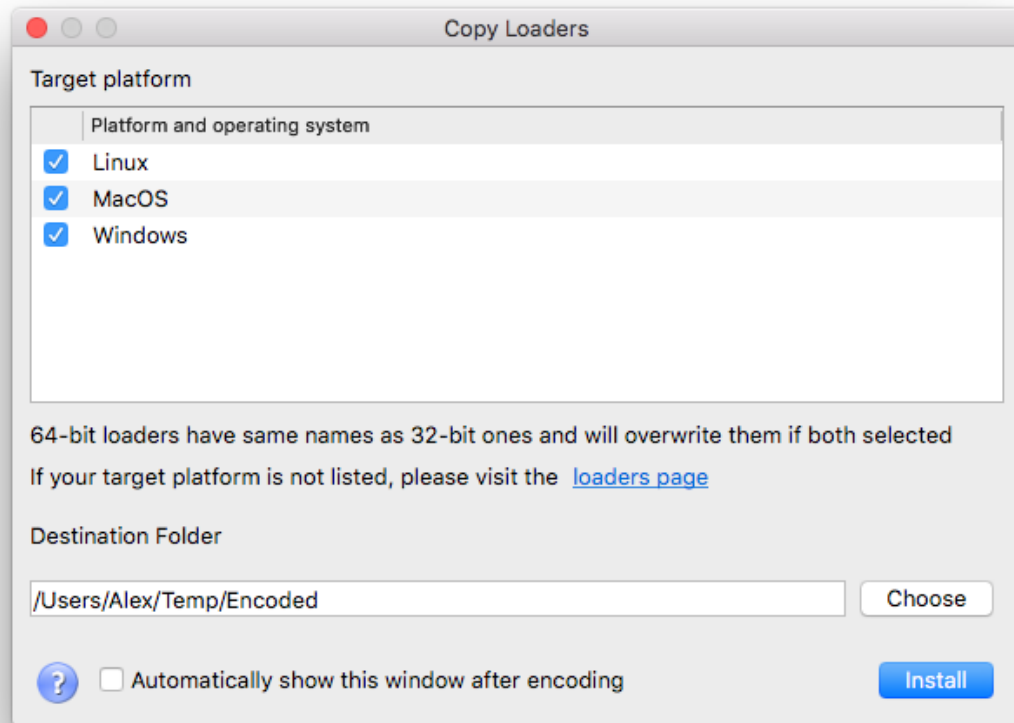
If you are running a web server locally on the same machine which runs SourceGuardian you may enter a local HTTP link like this `http://localhost/path/to/phpinfo.php` (specifying a real path to the phpinfo() page running in your local web server of course).

If your target machine is not available online you may copy the phpinfo() page contents from it and paste it to the large text box in this window. You may copy either HTML or plain text contents of the phpinfo() page. Then click "Suggest Me" to know details about how to download and install an appropriate loader for your system.

To display this "Install Loaders" window you may select File/Install Loaders menu item.

This window is automatically shown when the project has been successfully encoded. You may deselect a checkbox if you do not want this to happen every time you encode your project. You may restore this option in [File/Preferences](#) after it has been switched off.

2.7 Copying Loaders



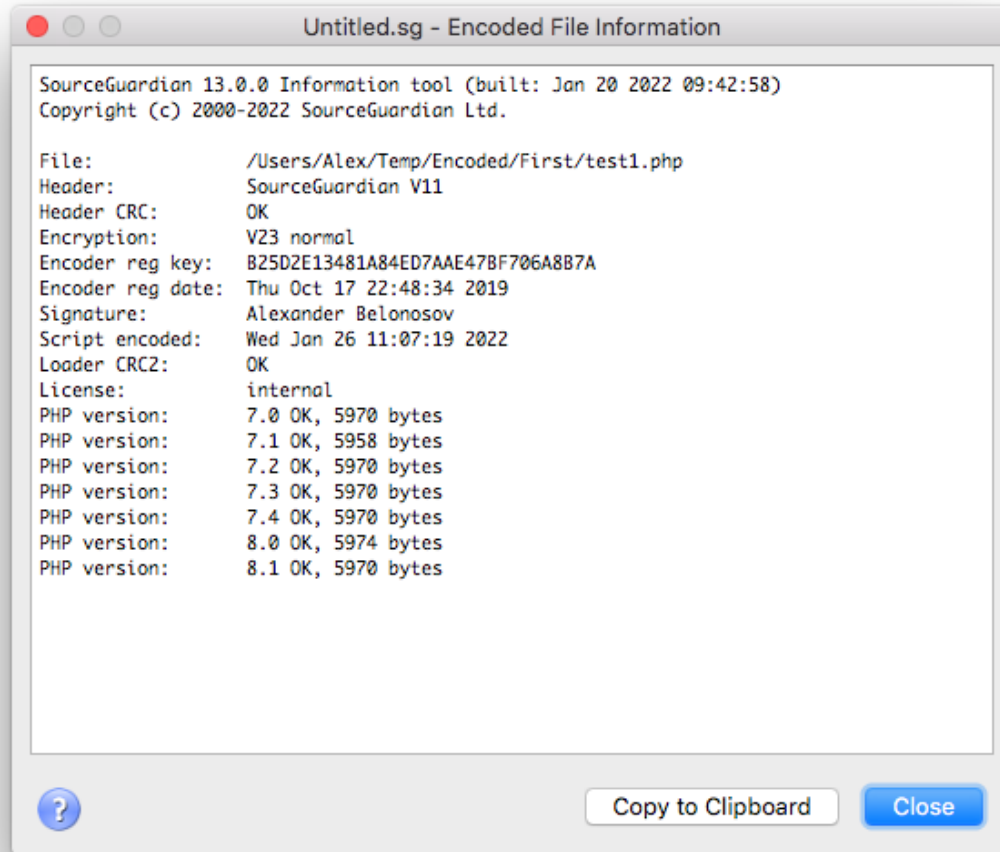
Loaders for some operating systems are prepackaged with the encoder. To copy loaders into encoded scripts folder select File/Copy Loaders menu item. Select loaders to install, choose a destination folder and click Install. Loaders will be copied to the /ixed/ subfolder within the destination folder.

Please note: Loaders for some target OS may have the same names and will overwrite each other if both selected. E.g. Loaders for i386 Linux and ARM Linux have the same names. Do not select such OS at the same time.

This window is automatically shown when the project has been successfully encoded. You may deselect a checkbox if you do not want this to happen every time you encode your project. You may restore this option in [Preferences](#) after it has been switched off.

Please note, we do not include loaders for all the supported platform to the GUI. Only loaders for main OS such as Windows, macOS and Linux are included. Also loaders included to the GUI are the versions actual by the release date of the GUI package. Sometimes we update the loader to fix the issues, please check our [blog](#) for further information about loaders update. You may download the recent versions of the loaders from our [loaders page](#).

2.8 Getting file information



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

Choose File Information from the File menu, then select a protected file or a license file from the dialog. Information about the protected script or the external license will be displayed.

It's possible to display script information only for files created **with the same installation** of SourceGuardian. If the script is locked to an external license file and the license file is also available then the information about the license will be included to the output. As Project ID and Project Key values are both required for getting the license information you may get this information only if you open the project that the license was generated from, and use File Information then. Project ID and Project Key values from the current opened project will be used for decoding the license information.

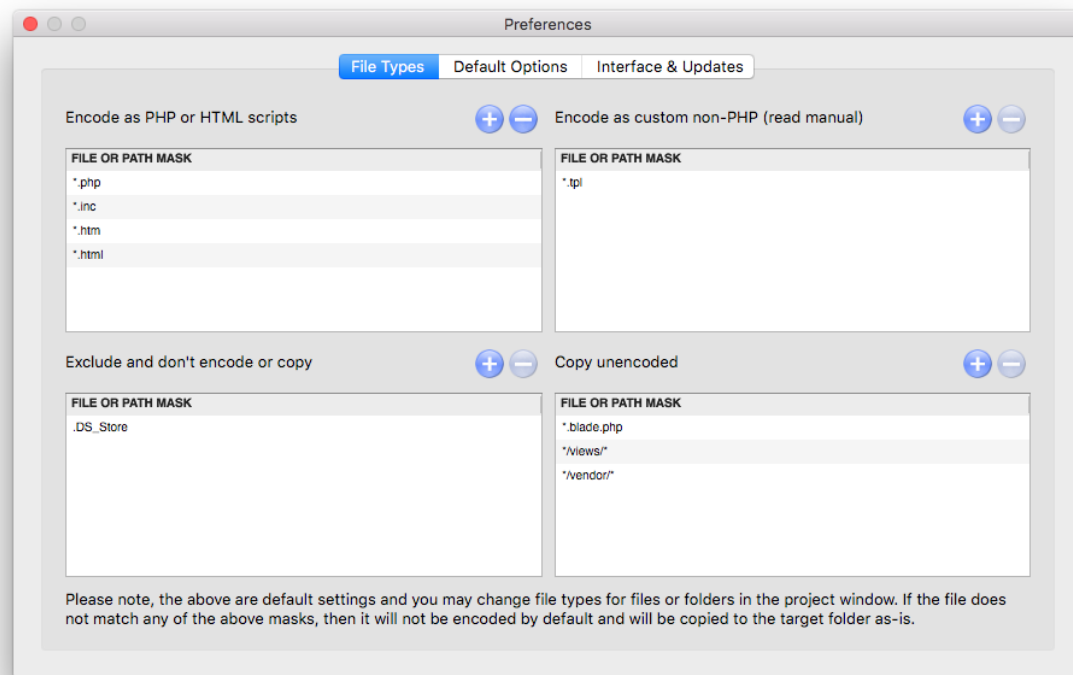
If you prefer to use the command line for getting file information, please use [sginfo command line tool](#).

2.9 Preferences and default settings

Choose Preferences from the File menu to display Preferences window.

macOS users - choose Preferences from the application menu or press Command + ,

File Types



File types tab allows you to set file types which will be used for setting default encoding mode when you add files to the project. Also you can specify file types which will not be added to a project when you add multiple files or folders. Use "Encode as PHP or HTML scripts" list to set file types for encoding using PHP script mode by default. Use "Encode as custom non-PHP" list to set file types for encoding in that special mode by default (see the [Project section](#) in this manual for further information about possible encoding modes). You may use * or ? wildcards.

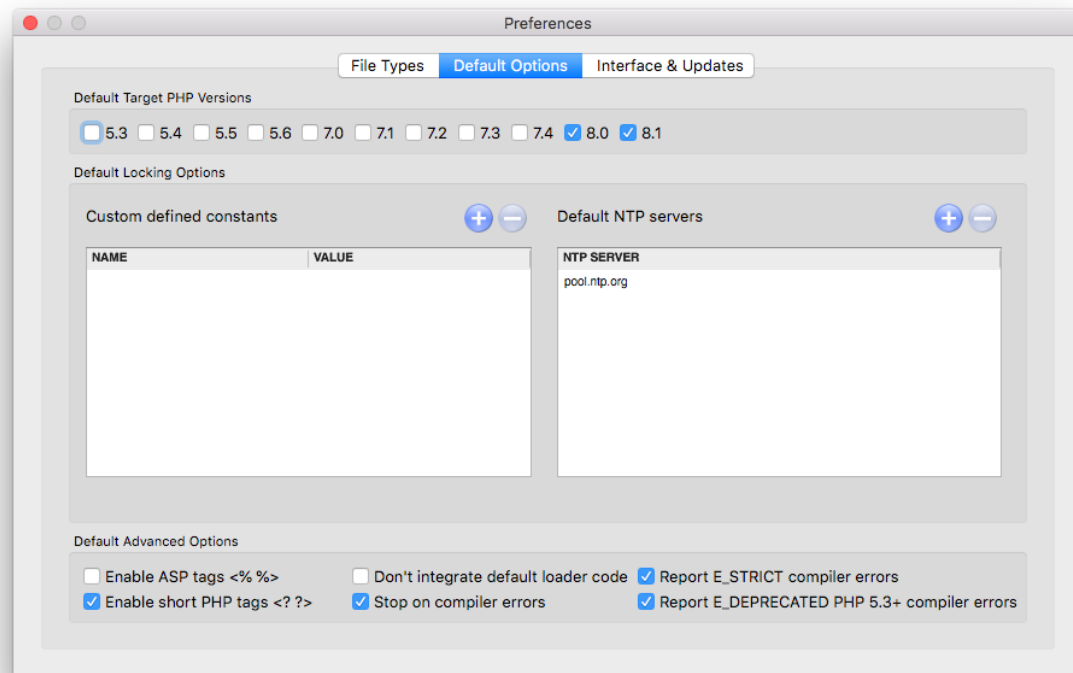
The "Exclude" list lets you specify files or folders that should not be added into the project when you add files or folders. When specifying files or folders to be excluded you may specify exact file names or use file masks (* and ? symbols may be used)

Use the "Copy unencoded" list to configure files which must be copied to the destination folder as-is without encoding. This is useful if you don't need to encode some of the files but still need them to be copied. They may be templates, configuration files, javascript, texts, images etc.

Default setting is to encode php, htm and html files as PHP scripts.

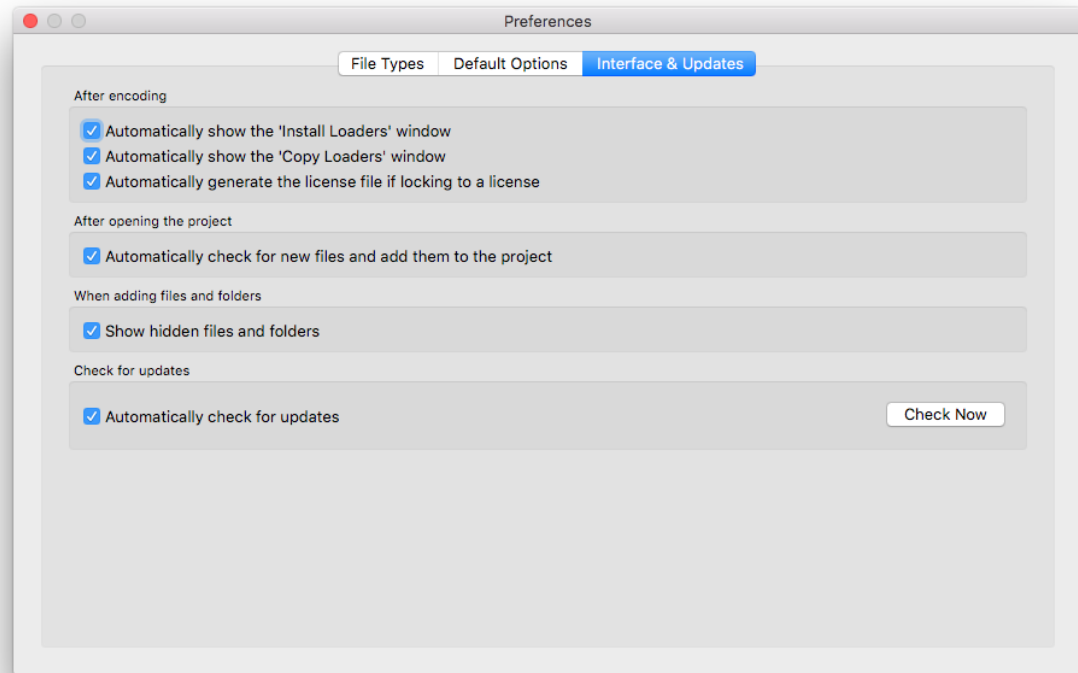
Please note, the above lists only define the default behavior and assigning encoding mode for newly added files. You may always change encoding mode per file or folder in the [project window](#).

Default Options



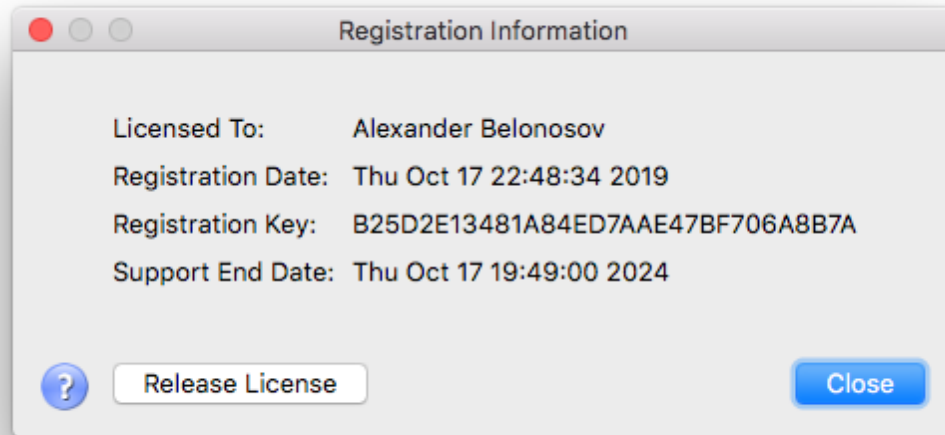
Project defaults tab allows you to set default encoding options which will be used for new projects. You may set default target PHP versions, define custom constants, define a list of online time servers you prefer to use for date checking, set default [advanced](#) options. This tab is useful for setting up default options to values you usually need for all your projects.

Interface & Updates



Use Interface & Updates tab to configure if "Loaders installation" window and "Install loaders" window will be shown after successful encoding and also other self descriptive settings. Use "Check for updates" option to enable or disable automatic checking for new versions of SourceGuardian. Click on "Check Now" to check for the update manually.

2.10 Viewing registration information



You may check your license information and the support end date by choosing Registration Information from the Help menu. Your name, registration date and support end date will be displayed. For evaluation version the expiry date is also shown.

You may release the current SourceGuardian license by clicking on "Release License" button. We will ask you for confirmation. Releasing the license lets you reinstall the encoder to another machine or to the same machine after upgrading hardware or OS. If you are going to upgrade the machine or OS, please firstly release the license and then you may transparently re-install your copy of SourceGuardian when the upgrade is complete.

You get 3 free license resets with the initial purchase. If you purchase an additional license or purchase a copy for another OS, each new license also gets 3 free resets. If you need to release the license after using all the 3 free resets, please [contact us in support](#).

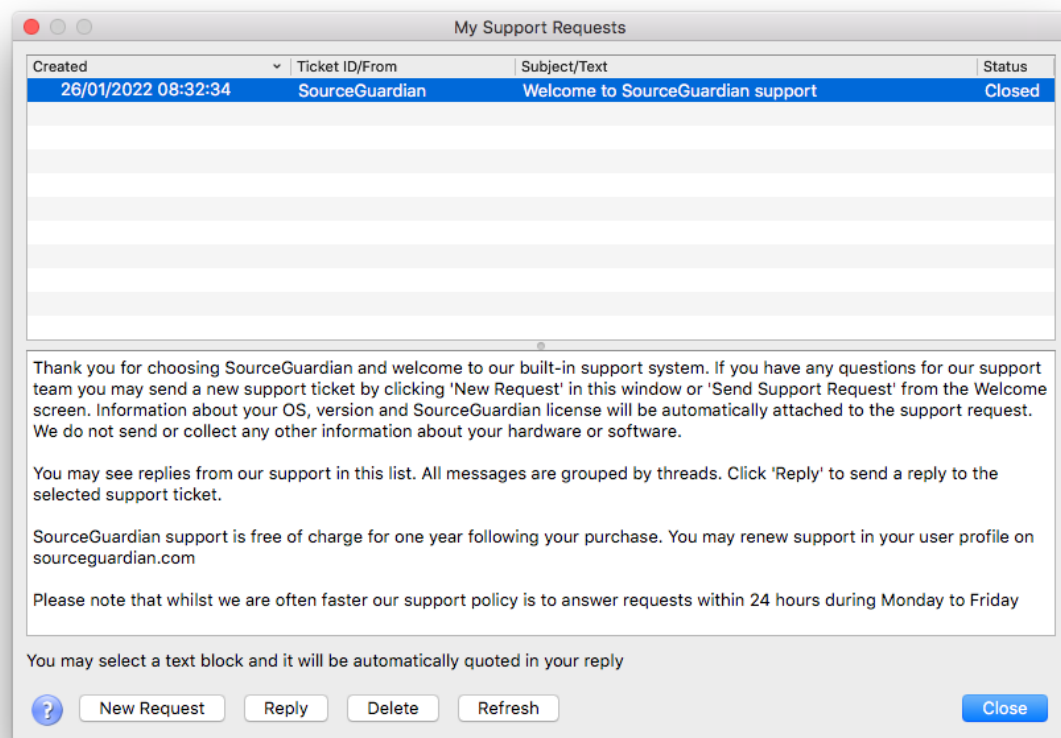
2.11 Getting help

Using the Help menu you can read built-in help, send a support ticket to our support team, download latest loaders and access our web site. Help is also always available by pressing Alt+F1 shortcut. Most windows within the application include a context help available by clicking on the ? (question mark) button.

If you have any questions about using SourceGuardian and could not find the answer in our manual or have any suggestions for our product feel free to use built-in support or contact us support@sourceguardian.com

Built-in support

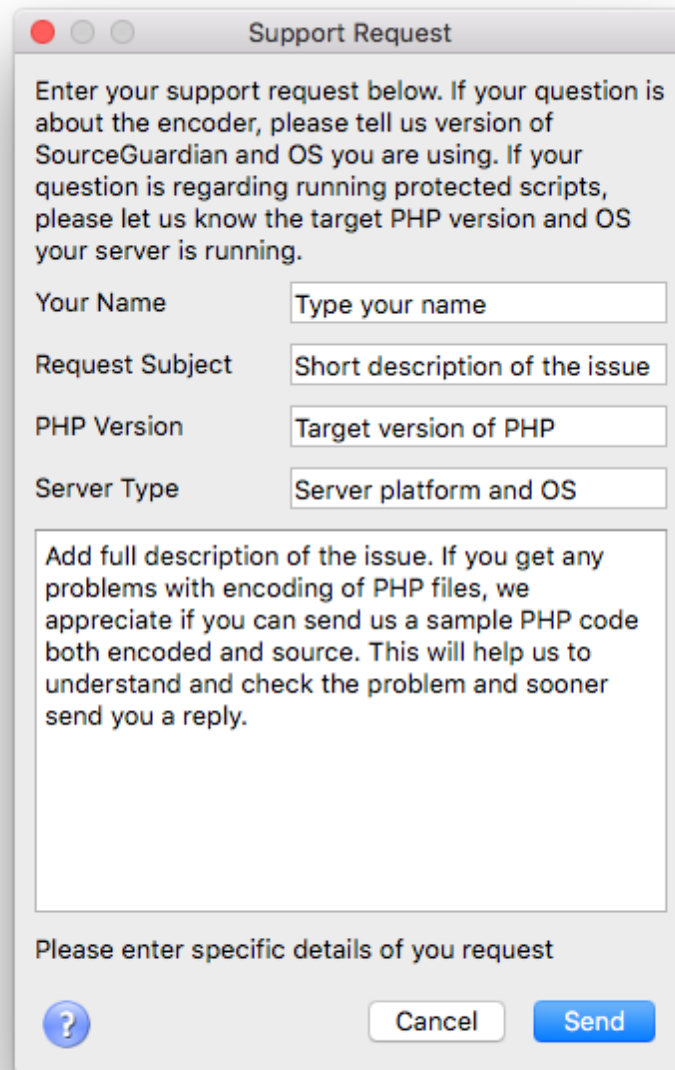
Since version 9 of SourceGuardian it includes built-in support which lets you easily send your questions to our support team, get answers and track issues - all in one window. Select "My Support Requests" from the Help menu to open built-in support window. You may also send a ticket directly without opening the tickets window by selecting "Send Support Request" from the Help menu. Built-in support is also available from the [Welcome screen](#).



The "My Support Requests" window displays a list of questions you ever sent to the support team and received answers. All the questions (tickets) are grouped by a "thread" which lets you easily find and track multiple opened questions or issues at the same time. Unread answers are displayed in bold. Click on the ticket header in the list to read its details below in the text box.

If you need to send a new ticket please click on "New Request". If you want to send a response, select a ticket and click on "Reply". The list is automatically updated. Your computer need to be connected to the Internet in order built-in support to work. You may click on "Refresh" to update the list at any time.

If you want to quote a text from the previous question or answer in the new response, please select the text in the text box and then click on "Reply".



A macOS-style dialog box titled "Support Request". It contains a text area for the request, four labeled input fields, a large text area for details, and a footer with a help icon, "Cancel", and "Send" buttons.

Support Request

Enter your support request below. If your question is about the encoder, please tell us version of SourceGuardian and OS you are using. If your question is regarding running protected scripts, please let us know the target PHP version and OS your server is running.

Your Name

Request Subject

PHP Version

Server Type

Add full description of the issue. If you get any problems with encoding of PHP files, we appreciate if you can send us a sample PHP code both encoded and source. This will help us to understand and check the problem and sooner send you a reply.

Please enter specific details of you request

? Cancel Send

When sending a support request please be as much specific as possible and send us all the details about your OS, version of SourceGuardian, your target OS, version of PHP, processor etc. Please do not forget to send us error messages that you get. Having all the details in your support request helps us to resolve the issue and quickly send you a reply.

New replies we sent to your support tickets will be displayed in "My Support Requests". Also you may see a "new" counter under "Support" in the [Welcome screen](#).

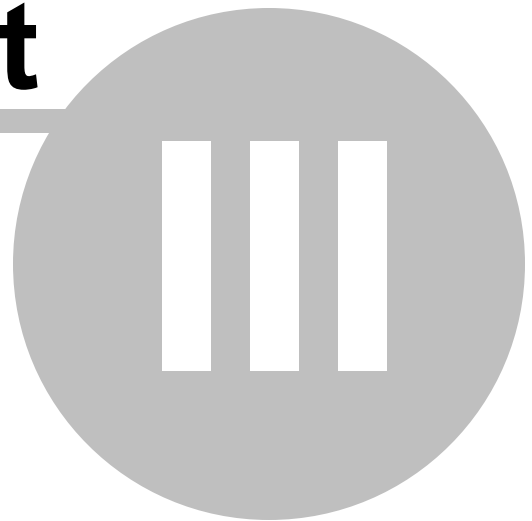
2.12 Checking for update

We keep working on SourceGuardian for PHP and release updates periodically. SourceGuardian does automatic updates by default and will inform you when a new version of SourceGuardian is available. You may also check for updates by click on "Check Now" in [Preferences](#). Automatic updates may be switched off in Preferences but we recommend you to leave this option on to keep your SourceGuardian installation up to date.

We update loaders when new official version of PHP is released. To check for loaders update please visit the loaders page on our web site by selecting Help/Download Latest Loaders or click to open this link in your browser <http://www.sourceguardian.com/loaders/>

SourceGuardian 13 for macOS, Windows, Linux

Part



3 Command line encoder

3.1 Command line tools installation

3.1.1 Windows

Installation for Windows is easy. Double click the downloaded SourceGuardian installer executable SourceGuardian-13.exe or sourceguardian-13-Evaluation.exe in order to run it. Follow instructions on the screen. SourceGuardian installer will install the software and then run the GUI encoder.

Command line encoder and tools are included with the GUI installation. You may find them within the SourceGuardian installation folder. Note, the command line encoder is named 'sgencoder.exe' to distinguish it from the GUI application.

If you have SourceGuardian installed to C:\Program Files a full path to the command line tools will be:

C:\Program Files\SourceGuardian 13\

You may run command line tools directly from the GUI installation folder. For convenience they automatically share the GUI encoder license and you may use both GUI and command line tools on the same machine under the same license.

3.1.2 Mac OS

Installation for macOS is easy. Double click on the downloaded DMG. When the DMG is opened, drag the SourceGuardian icon onto the Applications icon, which will install the SourceGuardian application into your Applications folder. Double click the SourceGuardian icon in Applications to run the encoder.

Command line encoder and tools are included with the GUI installation for macOS. You may find them within SourceGuardian.app application package. Right click the SourceGuardian icon in Finder, choose 'Show Package Contents', navigate to Contents/MacOS. Note, the command line encoder is named 'sgencoder' to distinguish it from the GUI application.

If you have SourceGuardian installed to /Applications a full path to the command line tools will be:

/Applications/SourceGuardian.app/Contents/MacOS

You may run command line tools directly from the GUI installation folder. For convenience they automatically share the GUI encoder license and you may use both GUI and command line tools on the same machine under the same license.

3.1.3 Linux

SourceGuardian for Linux is available in two packages - with or without GUI. You may choose one and download from your user profile. Both GUI and CLI packages are packed to tar.gz. For Linux we recommend that you install to /usr/local or your home directory /home/username.

Command line encoder and tools are included with the GUI installation for Linux. You may find them within the installation folder after installing the application. Note, the command line encoder is named 'sgencoder' to distinguish it from the GUI application.

The following instruction is expecting you use a terminal for installation. The downloaded installation

package may have the following names according to the OS, platform and version of the encoder:

SourceGuardian-13-Evaluation-Linux-x86.tar.gz
SourceGuardian-13-Evaluation-Linux-x86-CLI.tar.gz
SourceGuardian-13-Linux-x86.tar.gz
SourceGuardian-13-Linux-x86-CLI.tar.gz
SourceGuardian-13-Evaluation-Linux-x86_64.tar.gz
SourceGuardian-13-Evaluation-Linux-x86_64-CLI.tar.gz
SourceGuardian-13-Linux-x86_64.tar.gz
SourceGuardian-13-Linux-x86_64-CLI.tar.gz

Unpack the downloaded encoder installation file.

GUI installation example (note, tar.gz package file name may differ):

Copy the downloaded file to the destination directory /home/username. You may use either OS interface or terminal to locate and copy the downloaded file:

```
> cp /your/path/to/downloaded/SourceGuardian-13-Evaluation-Linux-x86_64.tar.gz /home/username
```

Unpack:

```
> cd /home/username  
> tar xzf SourceGuardian-13-Evaluation-Linux-x86_64.tar.gz
```

```
> cd SourceGuardian*
```

Run the GUI encoder:

```
> ./SourceGuardian
```

Optionally install icons to your Linux desktop (GNOME):

```
> ./install-menu-icons.sh
```

CLI installation example (note, tar.gz package file name may differ):

Copy the downloaded file to the destination directory /home/username. You may use either OS interface or terminal to locate and copy the downloaded file:

```
> cp /your/path/to/downloaded/SourceGuardian-13-Evaluation-Linux-x86_64-CLI.tar.gz /home/username
```

Unpack:

```
> cd /home/username  
> tar xzf SourceGuardian-13-Evaluation-Linux-x86_64-CLI.tar.gz
```

```
> cd sourceguardian*
```

Run the CLI encoder, proceed to automatic registration:

```
> ./sourceguardian
```

Run again after registration to see a brief list of the options:

```
> ./sourceguardian
```

The CLI installation package has the following structure:

sourceguardian-13/bin/sourceguardian

SourceGuardian executable

sourceguardian-13/bin/sgencoder5*.so	Internal encoder for PHP5+ (cannot be used directly)
sourceguardian-13/bin/sgencoder7*.so	Internal encoders for PHP7+ (cannot be used directly)
sourceguardian-13/bin/license.txt	license text
sourceguardian-13/bin/licgen	SourceGuardian Script License Generator (for full version only)
sourceguardian-13/bin/rinfo	SourceGuardian Information Tool (for full version only)
sourceguardian-13/loaders/*	Protected script loaders
sourceguardian-13/README	Startup document
sourceguardian-13/User_Manual.pdf	User manual in PDF format

When you run SourceGuardian for the first time we recommend that you proceed with automatic license registration. However, if your machine is not connected to the Internet, follow the [instructions below about encoder license installation](#).

3.1.4 Docker

Installing to a Docker container is experimental as of version 11.3 of SourceGuardian.

If you run the project deployment process from within a Docker container and need to install SourceGuardian to the Docker container, please get the SourceGuardian for Linux CLI packages. You may choose one and download from your user profile. Installation packages for Linux are available as tar.gz archive files.

The following instruction is expecting you use a terminal for installation. The downloaded installation package may have the following names according to the version of the encoder:

```
SourceGuardian-13-Evaluation-Linux-x86-CLI.tar.gz
SourceGuardian-13-Linux-x86-CLI.tar.gz
SourceGuardian-13-Evaluation-Linux-x86_64-CLI.tar.gz
SourceGuardian-13-Linux-x86_64-CLI.tar.gz
```

Please follow the instruction below in order to install SourceGuardian to a Docker container and then run it from there for encoding your files as a part of your project deployment process. Note, every installation of SourceGuardian to another machine or another Docker machine still requires an additional license as one license lets you install and use SourceGuardian only on one machine.

All the command below to be run in your Linux host console.

1) Download the SourceGuardian for Linux CLI installation package for your platform (x86 or x86_64) from your SourceGuardian user profile. Unpack to the home folder.

```
> cp /your/path/to/downloaded/SourceGuardian-13-Linux-x86_64-CLI.tar.gz /home/username
> cd /home/username
> tar xzf SourceGuardian-13-Linux-x86_64-CLI.tar.gz
```

2) Download a sample docker installation package from our website: <http://www.sourceguardian.com/knowledgebase/docker-installation.tar.gz> and unpack it to any local folder on your Linux host:

```
> cp /your/path/to/downloaded/docker-installation.tar.gz /home/username/
> cd /home/username
```



```
> tar xzf docker-installation.tar.gz
> cd docker-installation
> ls
```

This will show the following directory structure created in /home/username/docker-installation

```
.dockerignore
Dockerfile
README
docker-build.sh
docker-run.sh
entrypoint.sh
sourceguardian <empty folder>
```

3) Copy SourceGuardian encoder binaries to the sourceguardian subfolder within the docker-installation

```
> cp /home/username/sourceguardian*/bin/* /home/username/docker-installation/sourceguardian/
```

4) Edit ./entrypoint.sh and specify your SourceGuardian account email and password there

```
> vi entrypoint.sh
```

```
USERNAME=your@account.com
PASSWORD=yourpassword
```

4.1) For Mac users. Edit ./docker-build.sh and ./docker-run.sh and remove sudo from both commands.

4.2) For Mac users. Run Docker machine if it's not running.

```
>docker-machine start
>eval $(docker-machine env)
```

5) Run ./docker-build.sh to build a new docker image

6) Run ./docker-run.sh to run the image in the new container. It will register your copy of SourceGuardian and bind it to your Docker. Then it runs SourceGuardian from the Docker container for a test. A list of SourceGuardian CLI encoder options must be shown.

7) Add your files, check and edit ./entrypoint.sh again to add the encoding options, run again.

Please note, the method above maps your host's /var/run/docker.sock to make it available from within the container. See docker-run.sh. It is safe as you are running the deployment process on your development machine, Docker itself and the containers you are running on it are under your control.

If you have any questions, please email us: support@sourceguardian.com

3.1.5 First run

Please read and accept the SourceGuardian license during the first run of the encoder. Press the Enter/Return key for the next page, and if you agree with the license, type "I AGREE" and press the Enter/Return key on the last page when asked. It is our requirement that you accept the license terms in order to continue.

Since the SourceGuardian license is accepted, SourceGuardian will proceed to automatic registration if the Internet connection is available.

If automatic registration is not available, you will get a web link to our site and a hexadecimal registration code on the screen. Please visit our website

<https://www.sourceguardian.com/>

and login using the email and password we sent you during the online registration. If the email and password are correct you will be logged in to the user profile. Copy the hexadecimal registration code that was displayed earlier by the SourceGuardian application and paste it to the corresponding field under 'Available licenses'. Click 'Create License' and download the license file. If it does not start automatically, click on the 'Download' link.

Save the license (encode.lic) file and copy it into the command line encoder installation directory into bin/ subdirectory.

3.2 Running the command line encoder

SourceGuardian™ 13 for PHP command line encoder and tools are available along with GUI application for Windows, OSX, Linux. You need have an access to a terminal or any kind of remote shell in order to use it. Although SourceGuardian includes GUI application you may prefer to use a command line encoder for some reasons, for example for automatic encoding or license generation in your web site's backend.

3.2.1 Note for GUI users

If you are using command line tools preinstalled with a GUI version of SourceGuardian on macOS, Linux or Windows, please note, that the command line SourceGuardian executable is named 'sgencoder' instead of 'sourceguardian' in the manual below. This is to distinguish the command line encoder executable from the GUI executable which is also named 'sourceguardian'.

If you use a GUI version, type 'sgencoder' instead of 'sourceguardian' in all the commands you enter in the terminal. License generator and Information tool have their names unchanged in all the versions: licgen and sginfo accordingly.

SourceGuardian CLI package for Linux and SourceGuardian for FreeBSD have the encoder executable named 'sourceguardian' as it's mentioned below in the manual.

3.2.2 Usage

Running the command line encoder included to GUI

If you are running the command line encoder included with GUI, it automatically uses the GUI license. So you don't need a separate license to use the included command line tools on the same machine.

Name of the executable

If you are using the command line encoder which is included with GUI, the executable is *sgencoder*. If you are using a separate command line installation on Linux, the executable is *sourceguardian*. We are using the former in the samples, but if you are a Linux user and using a separate installation of SourceGuardian, please simply keep this in mind.

Specifying files to encode

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

You may run the SourceGuardian™ 13 for PHP encoder to encode either one or multiple files. 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 (file masks are supported and you may use '*' and '?' wildcard characters). If you use a file list its name must be prepended with @ in the command line.

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

A note for UNIX users: **Always quote masks on UNIX**, otherwise your shell interpreter will replace the specified file mask with real file and directory names and the result may be unexpected. You should always quote masks that specifies files to encode (like "*.php" in the example below) or in other options that accept file masks.

By default an encoded file will replace the original file. The original file will be backed up (unless you turn off the backup option with a -b- option). ".bak" extension is used by default for backups. We do not recommend that you turn off backups, make sure you have a copy of your source files!

We highly recommend that you always specify the output folder with -o (--output) option instead of encoding directly in the source folder(s). Backup are turned off in that case but source files remain unchanged and encoded files will be saved to the specified target folder. Make sure the source and target folders do not overlap.

--phpversion <version x.y>

SourceGuardian CLI encoder encodes scripts for all versions of PHP by default. This includes PHP4 which may be not compatible with you current PHP5 or PHP7 code. Use --phpversion option to specify versions of PHP you need to encode scripts for.

To encode a script to run under multiple versions of PHP, use the --phpversion option multiple times and specify all versions of PHP you need to run protected files under.

Your code must be compatible with ALL specified versions of PHP. Otherwise you will get an error message when encoding incompatible files and such files will remain unencoded.

```
--phpversion 4 - encode for PHP 4.x (4.3/4.4)
--phpversion 5.0 - encode for PHP 5.0/5.1
--phpversion 5.1 - encode for PHP 5.0/5.1
--phpversion 5.2 - encode for PHP 5.2
--phpversion 5.3 - encode for PHP 5.3
--phpversion 5.4 - encode for PHP 5.4
--phpversion 5.5 - encode for PHP 5.5
--phpversion 5.6 - encode for PHP 5.6
--phpversion 5 - encode for PHP 5.x (PHP 5.0-5.6)
--phpversion 7.0 - encode for PHP 7.0
--phpversion 7.1 - encode for PHP 7.1
--phpversion 7.2 - encode for PHP 7.2
--phpversion 7.3 - encode for PHP 7.3
--phpversion 7 - encode for PHP 7.x (currently PHP 7.0-8.1)
```

Example:

```
> sourceguardian file1.php
> sourceguardian --phpversion 4 --phpversion 5 file2.php file3.php file4.php
```

Optionally you may use "+" and "-" suffix for the --phpversion option. "+" means to encode for the specified version of PHP and for all the newer versions which are supported by the current version of SourceGuardian. "-" means to encode for all the supported versions of PHP except the specified one and all the lower versions, which is useful if you always need to encode by default for new versions but do not need support for old versions starting from some one. E.g.

--phpversion 4+	encodes for PHP4 and all the newer versions (up to 8.1 for this version of SourceGuardian)
--phpversion 5+	encodes for PHP5 and newer (up to 8.1 for this version of SourceGuardian)
--phpversion 7.1-	encodes for PHP 7.2 and newer, i.e. excludes PHP 7.1 and older

You may enumerate all the files you want to encode in a file list. 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 file list name in the command line.

Usage: >sourceguardian @filelist

When specifying a relative path don't start it with ../ or ./ directory specifiers.

It's possible to use shorter syntax for directory encoding. All specified directories will be recognized and the "*" file mask will be added:

```
>sourceguardian -r source_dir
```

which will work as if a file mask was specified:

```
>sourceguardian -r "source_dir/*"
```

A log will be printed to the terminal during the encoding process. A status message will be displayed for each encoded file. You may get one of the following status messages:

ok	The file was encoded without problem.
file not found cannot be read	The specified file could not be found. Check the specified file path.
PHP syntax or other compiler error	The original file has syntax or other errors and thus cannot be encoded. Check your file, test it with the PHP interpreter. This error usually appear when encoding for multiple versions of

	PHP and if your file is not compatible with all the target versions of PHP. Make sure your PHP script is compatible with all versions of PHP you are encoding for.
could not backup source file, skipped	The encoder could not make a backup copy of your original file (when no output directory was specified). SourceGuardian skips the file in that case to keep your original version. Check you have enough free space available and permissions to write to your original files directory.
cannot not write file	The encoder could not write the encoded file. Check you have enough free space available and permissions to write to original files directory or to the output directory if you have specified it with the -o option.
file is already processed by SourceGuardian	The encoder will not encode files which are already encoded with SourceGuardian. Check your original files directory.
empty file, skipped	The encoder will not encode empty files. If you need to have empty files for any reasons you may copy them manually.
not regular file, skipped	The encoder could not encode a file because it is not a regular file. It may be a socket or a unix device for example.
do not encode, skipped	The file was marked as 'do not encode' and therefore was skipped.
copied	The file was copied without encoding. It is possible when -f option is used to specify files to encode and -o option is used to specify output directory. All other files than specified in -f option will be copied as-is without encoding. It is useful for encoding an entire project directory when it also may contain non-PHP files.
internal encoder error, unknown error	This is an internal problem with the encoder. Check you have enough free memory space to run the encoder and some free space on the disk. If it is not a memory problem then let us know about this error. Send an email to support@sourceguardian.com with a detailed description of the error and the command line used for running the encoder. We will investigate the problem. We may also need some additional information from you.

3.2.3 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 the output directory and it differs from your source directory. The default backup option will be off when the output directory is specified. If you want to re-enable it, even when the output directory is specified, then use the -b <backup_extension> option after the output directory option.

Carefully specify the output folder which should never overlap with your source. The command line encoder does not do any checks for that.

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 directory with recursion and put encoded files to /

home/myproject/encoded.

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

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

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

Do not forget to quote file masks in the command line on Unix or Mac

3.2.4 Specifying which files to encode and which to copy

We have added an option into the command line encoder to specify which files should be encoded in PHP mode (-f). You may specify what files will be encoded specifying their filenames, filemasks or filelist. All other files which have been added for processing or found by expanding filemasks will be copied to 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:

```
>sourceguardian -r -f "*.php" -o "output_dir" ""
```

All (with recursion) *.php files from the current directory will be encoded and copied to output_dir. All other files from the current directory will be copied to output_dir as-is (unencoded).

You may specify multiple filenames or filemasks adding more than one -f option:

Example 2:

```
>sourceguardian -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.

You may also use the -c (--copy) filter option in addition to -f (--file) and -x (--exclude). The -c (--copy) option may be used to specify what files will be copied as-is without encoding to the target folder. This option makes sense only if you specify the target folder with -o (--output) option. If -c is used without -o, then it works as -x and skips the specified files. The option may take * and ? wildcards or a @filelist.

E.g. you may encode *.php files but keep *.tpl.php unencoded and copy the latter ones as-is:

```
>sourceguardian -o /path/to/target -r -f "*.php" -c "*.tpl.php" /path/to/source
```

Do not forget to quote file masks in the command line on Unix or Mac

3.2.5 Excluding files from processing

You may exclude some files or directories from processing. Please 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 * wildcard symbols). Wildcards in directory names are also supported. Note, excluded files will neither be processed, nor be copied to the target (-o) folder.

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

This will encode all *.php files in the current directory and all directories recursively but the 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 exclude, separated by a new line (masks are supported, use * and/or ? wildcard symbols). Specify the @ before the filelist name in the command line, e.g. `-x @excludefilelist`

Also it's possible to permanently mark files for skipping from encoding. [See details.](#)

Do not forget to quote file masks in the command line on Unix or Mac

3.2.6 Encoding entire directory contents

It's possible to use shorter syntax for directory encoding. All specified directories will be recognized and the "*" filemask applied.

```
>sourceguardian -r source_dir
```

instead of the following

```
>sourceguardian -r "source_dir/*"
```

3.2.7 Locking options (full version only)

Note for evaluation version users: all the script locking options are disabled for the evaluation version of SourceGuardian. However, you may read below about the available options and script locking features that work in the full version of the encoder.

--expire <dd/mm/yyyy>

Using 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".

--expire <00d[00h[00m[00s]]>

You can set the script to expire in a number of days/hours/minutes/seconds (from today). The script will not run after that time and will fail with the error message: "SourceGuardian Loader - the script has expired. Contact the script author about this problem. Error code [09]".

Example 1: `--expire 10d`

Example 2: `--expire 1d12h`

Example 3: `--expire 90m`

--time-server <server, server, ...>

If you use the --expire option you may also wish to let the script get the world time from the online time service for checking time rather than using the server time. Use --time-server option to specify time servers. You may specify multiple servers IP addresses or domain names separated with "," or ";"

NTP protocol is used, UDP on port 123 on remote server. NTP is always tried at first. TIME protocol is

the second, TCP on port 37 on remote server, checked if no response on NTP port.

If you specify a time-server option then your script will **require the Internet connection in order to run.** Time servers will be checked in the specified order. If neither of the servers from the list is available, the script will stop with the error:

"SourceGuardian Loader - the script requires an internet connection to run. The file has been encoded to run only when an internet connection is available. Setup an internet connection. Error code [20]"

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

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 the other included scripts.

A 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 the time-server option as usual for this but don't specify the expiration date for the script. The script will try to access the online time service and will fail if it's offline and hence not possible to read from the time server.

--domain <domain>

Bind the scripts 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 from any other non-authorized domain, the script will fail with the error message: "SourceGuardian Loader - the script is not licensed to run on this machine or it is not running in a web server environment. Run this script in a web server environment. Contact the script author about this problem. Error code [02]". You may use this option more than once to specify multiple domains. This option may not be used with the --domain-encrypt option.

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

Please note, the loader is able to check the domain name ONLY if the protected script is running in a web server environment. The web server must also provide the PHP interpreter with one of the following environment variables; SERVER_NAME or HTTP_HOST. The Apache web server does it by default and the variables are available in PHP from \$_SERVER superglobal array. For the Nginx webserver you may need to have an appropriate fastcgi_param line in the web server configuration file to make that variable be passed to the PHP script. For other web servers please find and use appropriate configuration directives.

As a developer you may wish to check if the SERVER_NAME or HTTP_HOST variable is available in the environment on the target server by investigating \$_SERVER['SERVER_NAME'] or \$_SERVER['HTTP_HOST'] superglobals.

Note, if you are running your encoded PHP script with PHP CLI, e.g. as a cron job, the above mentioned SERVER_NAME and HTTP_HOST variables will not be set. This means if you lock scripts to a domain, they won't run from CLI as the domain name information will not be available to the loader and it can't validate it. But read below about --domain-ignore-cli option if you need to run your protected files with PHP CLI.

Hint: use the name of the main domain in this option, not the name of any sub domain unless 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 the script with a domain name. The encoder will lock the script to run only from the specified domain. The encoder will also use a specified domain name as a part of the key for encryption for enhanced security. The loader will not be able to decrypt a script in it's run from the invalid domain and then the script fails with the error message: "SourceGuardian Loader - Protected script's checksum error. Probably the encoded file was modified. If this script requires a license file in order to run, this error may be caused by an invalid license file. Please install an original unmodified file or contact the author of the script to get the original file or a license file. Error code [12]". You may use this option ONLY ONCE in the command line. This option may not be used along with the other --domain options.

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 you to run the script ONLY from mydomain.com and not from www.mydomain.com and vice versa.

You should not use wildcards for domain name when using this option. This option works ONLY for one strictly specified domain name.

Please read the above comments about SERVER_NAME and HTTP_HOST environment variables. This is also related to --domain-encrypt option.

--domain-ignore-cli

Locking scripts to the domain name will not let it run with PHP CLI as the domain name information is not available in CLI and the loader can't validate it. However, you may use the --domain-ignore-cli option to ignore the domain check for the scripts running with CLI PHP. It means if you have encoded the entire project with locking to a domain and selected the option to ignore the domain check for CLI, your protected files will require to be run under the specified domain(s) for web but the same files will be run OK with CLI PHP.

This simplifies running PHP CLI scripts like cron jobs but still have them encoded in the same project along with web scripts.

However, you may need consider the risk of using this option as it does what it does - lets run your domain locked encoded files with PHP CLI bypassing the domain check. Consider using the new [--remote-verification-url](#) option as a safer solution.

--ip <x.x.x.x{/y.y.y.y}>

Bind the scripts to 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 a real IP address before comparing to the target IP. So you may use this option to lock the script to multiple IP addresses if a mask is specified. If the script is run from any invalid IP address, the script will fail with the error message: "SourceGuardian Loader - the script is not licensed to run on this machine or it is not running in a web server environment. Run this script in web server environment. Contact the script author about this problem. Error code [01]" You may use this option more than once to specify multiple IP/mask pairs. IP address mask 255.255.255.255 is applied by default if not specified. This option may not be used with --ip-encrypt option.

Please note, the loader is able to check the domain name ONLY if the protected script is running in a web server environment. The web server must also provide the PHP interpreter with one of the following environment variables: SERVER_ADDR or LOCAL_ADDR. The Apache web server does it by default for PHP. For the Nginx webserver you need to have an appropriate fastcgi_param line in the web server configuration file to have that variable passed to the PHP script. For other web servers please find and use appropriate configuration directives.

As a developer you may wish check if SERVER_ADDR or LOCAL_ADDR variable is available in the environment on the target server by investigating \$_SERVER['SERVER_ADDR'] or \$_SERVER['LOCAL_ADDR'] globals.

Note, if you are running your encoded PHP script with PHP CLI, e.g. as a cron job, the above mentioned SERVER_ADDR and LOCAL_ADDR variables will not be set. This means if you lock scripts to IP, they won't run from CLI as the server IP information will not be available to the loader and it can't validate it. But read below about --ip-ignore-cli option if you need to run your protected files with PHP CLI.

--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 also use the specified IP address with applied mask as a part of the key for encryption for enhanced security. The Loader will not be able to decrypt the script running from the invalid IP address and will fail with the error message: "SourceGuardian Loader - Protected script's checksum error. Probably the encoded file was modified. If this script requires a license file in order to run, this error may be caused by an invalid license file. Please install an original unmodified file or contact the author of the script to get the original file or a license file. Error code [12]". You may use this option ONLY ONCE in the command line. IP address mask 255.255.255.255 is used by default if not specified. This option may not be used along with other --ip options.

Please read the above comments about SERVER_ADDR and LOCAL_ADDR environment variables. This is also related to --ip-encrypt option.

--ip-ignore-cli

Locking scripts to the IP addresss will not let it run with PHP CLI as the domain name information is not available in CLI and the loader can't validate it. However, you may use the --ip-ignore-cli option to ignore the IP address check for the scripts running with CLI PHP. It means if you have encoded the entire project with locking to IP and selected the option to ignore the IP check for CLI, your protected files will require to be run under the specified IP(s) for web but the same files will be run OK with CLI PHP.

This simplifies running PHP CLI scripts like cron jobs but still have them encoded in the same project along with web scripts.

However, you may need consider the risk of using this option as it does what it does - lets run your IP locked encoded files with PHP CLI bypassing the IP check. Consider using the new --[remote-verification-url](#) option as a safer solution.

--mac <xx:xx:xx:xx:xx:xx>

Bind the scripts to a LAN hardware (MAC) address. (Note, the "MAC address" name here is Media Access Control address and it is not related to Macintosh computers. All the computers connected to the Internet or LAN internally use MAC addresses). A MAC address is unique for the networking adapter and so it may be used for machine identification. A MAC address is 6 bytes long, with each byte represented in hex and separated with a ':' symbol. The encoder will lock scripts to run only from the machine which has a networking adapter with the specified MAC address. If there is more than one network adapter installed then the script checks all of them. If the script is run from another machine, the script fails with the error message: "SourceGuardian Loader - the script is not licensed to run on this machine. Contact the script author regarding this problem. Error code [03]" You may use this option more than once to specify multiple MAC addresses.

Hint: use 'ifconfig -all' command on macOS, Linux or FreeBSD or 'ipconfig /all' on Windows to get a list of installed networking adapters and known MAC addresses. On Macs you may find LAN hardware addresses in System Preferences/Network/Advanced/Ethernet/Ethernet ID.

Alternatively you may use `sg_get_mac_addresses()` API method and get an array filled in with MAC addresses detected on the machine where you run this method. It may be called directly or from the encoded PHP code (not locked with any options). Note, as this API method is binary compiled into the loader, an appropriate SourceGuardian loader but me installed and loaded (require directive) before your code calls this method. We recommend that you create a mini project, encode it and include the loaders for obtaining MAC addresses from the client's machine, in that case loaders will be found automatically.

Locking the script to a LAN hardware address may be the only option to lock the script to a machine when the script is not running in a web server environment or some environment variables are not available for server IP address or server domain name check. E.g. running a script as cron task or a command line script.

--machine-id <machine id>

Bind the scripts to a machine ID. Machine ID is a unique identifier of the computer where your protected files are run. We use a special approach to know the machine ID and it differs for the platforms where SourceGuardian loader is installed. Machine ID is a hash represented as a hex code. The encoder will lock scripts to run only from the machine which has the same machine ID as specified in the option. If there are more than one --machine-id option used, then the protected code will run on any of them. If the script is run from another machine, the script fails with the error message: "SourceGuardian Loader - the script is not licensed to run on this machine. Contact the script author regarding this problem. Error code [04]"

Use `sg_get_machine_id()` API method and get the machine ID of the computer where you run this method. It may be called directly or from the encoded PHP code (not locked with any options). Note, as this API method is binary compiled into the loader, an appropriate SourceGuardian loader must be

installed before your code calls this method. We recommend that you create a mini project, encode it and run for obtaining the machine ID from the client's machine before locking to it.

Locking to a machine ID is an alternative to using MAC addresses locking for the scripts which are not running in a web server environment. E.g. running a script as cron task or a command line script.

--machine-id-encrypt <machine id>

Bind and encrypt to a machine ID. The encoder will lock the script to run only on the machine with the specified machine ID. The encoder will also use the specified machine ID as a part of the key for encryption for enhanced security. The Loader will not be able to decrypt the script running from the invalid machine and will fail with the error message: "SourceGuardian Loader - Protected script's checksum error. Probably the encoded file was modified. If this script requires a license file in order to run, this error may be caused by an invalid license file. Please install an original unmodified file or contact the author of the script to get the original file or a license file. Error code [12]". You may use this option ONLY ONCE in the command line. This option may not be used along with other --machine-id options.

--external <filename|path|URL>

Binds the scripts to an external license file. The scripts will require the license file in order to run. This file may be deployed along with the script or separately. This option lets you encode your scripts once and deploy to your clients with different licenses. Every license may have a different number of locks applied to it. You should specify only an external license file name here. Example: --external mylicense.lic No real license file will be created during encoding. Use SourceGuardian licgen tool or GUI for creating a license file. When running protected scripts, and there is no specified license file found, the script fails with the error message: "SourceGuardian Loader - This protected script requires ... license file in order to run. Please contact the author of the script to get a license file. Error code [13]" You may use this option only ONCE in a command line. This option may not be used with any other locking options.

Since your code is bound to a license file, you may use locking options for the licenses rather than the encoded files. Therefore, the encoded files may remain the same for multiple clients while licenses (and locking options applied to them) differ per client.

If the name of the license file does not include full path then protected scripts will search for the license file in the protected script's folder, its parent folder and so on. So you may have one license file for the entire project located in the top project's folder. A path may be specified for a license file. It may be an absolute file system path or URL like <http://xxx/mylicense.lic> (you may use any name for the license file). If the protected script cannot find the specified license file it displays the error message: "script requires ... file to run".

Example: --external script.lic

Example: --external <http://myserver.com/user123.lic>

Multiple external license file names may be specified when locking to a license. Separate license file names, paths or URLs by a comma in the --external or an appropriate GUI option. Please note, the license files will be checked in the order they are specified. You may mix local file names, paths or URLs. If neither of the license files is found and running of the protected file fails with a "a license file is required" error, the error message will include only the first license file name. This is normal and by design.

The algorithm which is used for locking scripts to an external license gives your scripts a 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 is the most powerful and flexible way to protect your scripts. We recommend that you use external license file locking for all your scripts. Using of locking to a license file is also the best way of deploying the encoded project to different clients as you may encode once and then assign different restriction options for the clients in the licenses you generate per client.

Please read further details in the [Using external script license generator](#) section of the manual.

--projid <value>

Specify a Project ID to identify your project. Project ID must be specified for the --external option. When you use the licgen tool for creating a license file, it is important to specify the same Project ID that was used for encoding.

--projkey <value>

Specify the Project Key to encrypt your project. Project Key must be specified for the --external option. When you use the licgen tool for creating a license file, it is important to specify the same Project Key that was used for encoding

The encryption algorithm uses the idea of two keys. The first key (Project Id) is stored within the encrypted area of protected scripts and used to decrypt an external license file. The second key (Project Key) is stored within the license file and used to decrypt the bytecode from the protected script. This algorithm protects your product against creating a full working copy from the demo version by some people who may be interested in this. In order to decrypt and run the protected script a valid license file for the full version of your product is required. Otherwise it's impossible to decrypt and run the bytecode. Both Project Id and Project Key values are required if --external option is used.

--remote-verification-url <URL>

This option lets you use a special approach for protecting and locking your PHP CLI scripts. Using of this option is preferred if your CLI script is a part of your encoded PHP WEB project. In that case this option may be used for locking CLI scripts to run only on the same machine where your WEB part of the project is installed and run. A CLI script encoded with --remote-verification-URL will not run on another machine and will fail with the error message: "SourceGuardian Loader - the script is not licensed to run on this machine. Contact the script author regarding this problem. Error code [05]"

For this option to work you need to do two things:

1) Create a special encoded PHP script which is accessible via HTTP protocol, it will be used to validate the machine and return the verification ID to the CLI script. The only directive you need to put into it is:

```
<?php
    echo sg_get_verification_id();
?>
```

Note 1: if you use any frameworks, you may need to use another mechanism for sending a string to the

output stream instead of *echo*

Note 2: as `sg_get_verification_id()` API method is binary compiled into the loader, make sure you have encoded the verification script.

Normally, you will add this to your WEB part of the project and encode it with SourceGuardian along with the other web PHP files. You are free to use any name for this validation script. *Example:* `verification_id.php`

2) When encoding your CLI script use `--remote-verification-url` option to specify the full URL to your web verification script and use the same project ID as for the web part of your code and particularly the verification script.

Example: `--remote-verification-url http://mydomain.com/verification_id.php`

Note 1: as the `--remote-verification-url` is mostly used only for locking of the encoded CLI scripts, consider separate encoding of the web part of your project and CLI scripts. You may create two GUI projects for that or use the command line encoder twice. Use the same Project ID/Project key for both!

Note 2: You may use standard locking to IP or domain for your web verification script (as usual for this to work your web server must send the information about current IP and domain to PHP via environment). Anyway, you "lock" to the domain if use the domain name in the verification URL or lock to IP if you use IP in the URL - choose the way which suits your project and the deployment process.

Note 3: you should not worry about the delays HTTP adds, they are minimal as your CLI code and the main WEB code are both running on the same machine. However, it's recommended to check the verification URL is accessible from CLI, e.g. using 'curl' or 'wget'. If the domain name can't be recognised from CLI, you may add the domain name to hosts file or switch to using IP instead of the domain name.

However, if your CLI PHP script works on its own and is not a part your your web based project, then you still may use locking to a machine ID described above as well as good old locking to MAC addresses.

--conj

This option makes sense only when encoding multiple files. All scripts encoded with this option will work only with other encoded files of the same project and will NOT work if any of the included files or top files are substituted with an unencoded one or encoded as a part of another project or 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. Enabling this option makes it impossible to substitute `c.php` with their own code and do `'echo $password'` to know your password. Also enabling this option makes it impossible to create `d.php` which includes protected `a.php` and then does `'echo $password'`.

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 that you always use this feature if your project has any secure data embedded to the

scripts such as usernames, password, database names etc.

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

Note: this option must be applied if you use [sourceguardian.restrict_unencoded](#) = "1" in php.ini

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 if forgotten. They are required for generating licenses for your customers.

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

3.2.8 Advanced options

--asp-tags

Enables use and recognition of ASP-like `<% %>` tags for indicating the PHP code in addition to standard `<?php ?>` tags.

--no-short-tags

Disables use and recognition of short PHP tag `<?>` for indicating the PHP code. Both standard `<?php ?>` and short `<? ?>` tags are recognized by default.

-n

Don't integrate default starter code. You may use this option if you don't want to include the default starter code into protected scripts. Scripts encoded using this option will not be able to automatically find and load an appropriate SourceGuardian loader and you have to install the ixed loader manually to run this script. See this [section](#) about the manual ixed installation. If you already have the SourceGuardian loader installed server-wide in php.ini then this option may be useful.

Since PHP 5.2.5 dynamic extensions including SourceGuardian loaders must be installed to PHP's `extension_dir` folder specified in the php.ini configuration file and an appropriate `extension=ixed.XX.YYY` directive must be added to the php.ini in order to install the loader.

Note: if you select this option then "Loader not found error code" (-j) option has no effect (as the code is placed inside the default starter code).

-p "code"

Custom header. This options lets you add a custom header at the top of every encoded file. You may put any code to be executed BEFORE the protected scripts code. This code WILL NOT BE ENCODED

(although it is still protected with CRC against changes). This may be either HTML text or PHP code. For PHP code - you should enclose your code with `<?php ?>` tags. This option is usually used for including copyrights into protected scripts but also it's used for including [custom error handler](#) functions. 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\"; ?>"
```

You may load the contents of a custom header from the file. Replacing double quotes is not needed in that case.

Example 4:

```
-p @my_custom_header.php
```

-j "code"

Change loader not found code. It is possible to change the default action when an appropriate SourceGuardian loader is not installed and could not be found or used for automatic dynamic loading. The default handler included into protected script starter's code (which prepends each protected script by default) will display an error message "This script is protected by SourceGuardian™ and requires file ... " and then the script stops executing. This option allows you to change the default error action. You may use any HTML text or PHP code and it will be displayed or executed as a replacement to the default SourceGuardian™ loader error. This code **WILL NOT BE ENCODED** (although it is still protected with CRC against changes). This may be either HTML text or PHP code. For PHP code - you should enclose your code with `<?php ?>` tags. 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(); ?>"
```

You may load the contents from a file. Replacing double quotes is not needed in that case.

Example 3:

```
-j @my_loader_not_found.php
```


Example 4:

```
>sourceguardian -j "<?php echo \"Loader is not installed. Call 123-456-7890 for support\"; exit(1); ?>\"
hello.php
```

The encoded script will have the defined code as unencoded:

```
<?php
if(!function_exists('sg_load')){$_v=phpversion();$_x=explode('.',$_v);$_v2=$_x[0].'(int)$_x
[1];$_u=strtolower(substr(PHP_UNAME(),0,3));$_ts=(@constant('PHP_ZTS') || @constant
('ZEND_THREAD_SAFE'))?'ts:.';$_f=$_f0='ixed.'.$_v2.$_ts.'.$_u;$_ff=$_ff0='ixed.'.$_v2.'.(
(int)$_x[2]).$_ts.'.$_u;$_ed=@ini_get('extension_dir');$_e=$_e0=realpath
($_ed);$_dl=function_exists('dl') && function_exists('file_exists') && @ini_get('enable_dl') && !
@ini_get('safe_mode');if($_dl && $_e && version_compare($_v,'5.2.5','<') && function_exists
('getcwd') && function_exists('dirname')){$_d=$_d0=getcwd();if(@$_d[1]!=':') {$_d=str_replace(
'\','/',substr($_d,2));$_e=str_replace('\\','/',substr($_e,2));$_e.=(($_h=str_repeat('/.',substr_count
($_e,'/')));$_f='/ixed/'.$_f0;$_ff='/ixed/'.$_ff0;while(!file_exists($_e.$_d.$_ff) && !file_exists
($_e.$_d.$_f) && strlen($_d)>1){$_d=dirname($_d);if(file_exists($_e.$_d.$_ff)) dl
($_h.$_d.$_ff); else if(file_exists($_e.$_d.$_f)) dl($_h.$_d.$_f);if(!function_exists('sg_load')
&& $_dl && $_e0){if(file_exists($_e0.'/'.$_ff0)) dl($_ff0); else if(file_exists($_e0.'/'.$_f0)) dl
($_f0);if(!function_exists('sg_load')){$_ixedurl='http://www.sourceguardian.com/loaders/download.
php?php_v='.$urlencode($_v).'&php_ts='.(($_ts?'1':'0')).'&php_is='.@constant('PHP_INT_SIZE').'&os_s='.
urlencode(PHP_UNAME('s')).'&os_r='.urlencode(PHP_UNAME('r')).'&os_m='.urlencode(PHP_UNAME('m')); ?>
<?php echo "Loader is not installed. Call 123-456-7890 for support"; exit(1); ?><?php exit();}}return
sg_load('706A8B7A65CF5EE6AAQAAAAAXAAABHgAAACABAAAAAAAAD/
wgTlbsGhJLb4FLHhSZ3U9ufJ9aBUMGlcrCUgz6cYgRWQtSwt932887i4gXG+xVA+2gkIYpCkVIlTWqhz
LBRTiHDxejyW7imMx65J7Alf0RHm1rQyGVQS0cwF5mh+5WaeBljpG6pcl2UnhT7
+RIGzTe7dmeg8bHJSQAAAAKAAAABbMouU4GsE2Y+bugvgoKydm8Oc9fcaKA4usVcazxOP0TFyHd/
QwNno28hyaL+I7
+8blACHGBU5nL0a7QxU5s3zcl9PaDYmKqCicSwcn2jTH/4B3wkf4zFSFma7iZBuhJBg80HTXUcjr/4e52
GzRo43S7IKddTZ8sZiGEquFyUIBtvcpnFtIDKtU5n8Tpppq/gMVbDCHSgfig+KgQuLL/AAAAAA==');
```

Now a test run without a required SourceGuardian Loader installed:

```
>php -f hello.php
```

```
Loader is not installed. Call 123-456-7890 for support
```

Custom header code and Loader error code may be loaded from a file

Custom 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: sgencoder -p @prepend.php -j @loadererr.php myscript.php

The code is loaded during encoding and stored as is *non encoded* because that code is executed when no SourceGuardian loader is loaded or when protected script's bytecode is not decoded yet.

-z[0-9]

Specify compression level. Higher compression level gives smaller output scripts which run faster but encoding process will be slower (and vice versa).

--strict-errors

Report E_STRICT compiler errors. E_STRICT "Strict Standards" warnings were introduced in PHP 5. This option instructs the encoder to warn of such messages during encoding. This option is ignored when encoding scripts in PHP 4.x mode. Usually E_STRICT warnings may be ignored but it may be a good idea to let the encoder display such warning messages and review the code. The encoder will stop if --stop-on-error option is also specified.

Note: the encoder can catch only compiler-related E_STRICT warnings. Run-time E_STRICT messages will be displayed when the protected script runs as usual according to the error_displaying option in the php.ini.

--deprec-errors

Report E_DEPRECATED PHP 5.3+ compiler errors. E_DEPRECATED warnings were introduced in PHP 5.3. This option instructs the encoder to warn of such messages during encoding. This option is ignored when encoding scripts for PHP older than 5.3. Usually E_DEPRECATED warnings may be ignored but it may be a good idea to let the encoder display such warning messages and review the code. The encoder will stop if --stop-on-error option is also specified.

Note: the encoder can catch only compiler-related E_DEPRECATED warnings. Run-time E_DEPRECATED messages will be displayed when the protected script runs as usual according to the error_displaying option in the php.ini.

--stop-on-error

Stop on compiler errors. This option instructs the encoder to stop encoding at first critical error or E_STRICT/E_DEPRECATED warning if appropriate options are selected. This may be useful if you have many files in the project and there is a risk of missing errors and leaving some files unencoded because of it.

Note: Even if this option is off error messages will be printed to console during encoding.

--eval-compatible

Enables eval() compatibility for encoded scripts. Normally encoded scripts cannot be run with the PHP eval() function. Additional CRC check restricts it as the protected code is passed as a string and source file is unknown. This improves security for encoded scripts that run in a standard way. However, some PHP template engines or custom code requires loading encoded PHP scripts as a string and then passing it to the eval(). In order to enable running protected scripts with eval() you may use this option and encode those files in the 'eval compatibility' mode.

--keep-file-date

This option instructs the encoder to keep the modification date for encoded files the same as modification date of source files. This may help in deploying only updated files and in some other cases of custom deployment of encoded files. The modification date for encoded files is set to the current date by default if this option is not used.

--keep-doc-comments

PHP doc comments are removed by default. Use --keep-doc-comments option to keep them e.g. if the framework you use require doc comments. We added an appropriate tickbox to the [Advanced options](#) window in GUI.

--ignore-symlinks

Ignore symlinks during recursive folders scanning.

-a <YYYYMMDDhhmmss>

This options lets you encode only changed files detected by file modification date. Only files that have been modified after the specified date will be encoded.

3.2.9 Custom predefined constants

--const name=value

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

It is important to use quotes if your constant name or the value contains a space or other special symbols. You defines only one constant with one

--const option but you may add as many --const options as you need into the command line.

```
>sourceguardian --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. Add as many --const options as you need into the command line.

To get a predefined constant value from the protected PHP code use `sg_get_const()` API function. This function is defined in the SourceGuardian loader. `sg_get_const()` returns a predefined SourceGuardian constant value or FALSE if a constant with the specified name is not defined. SourceGuardian constants **names are case sensitive**.

If `sg_get_const()` is called without parameters, it returns an array of all the constants including standard ones.

Syntax: `string sg_get_const(string)`

There are 5 predefined constants for all protected scripts:

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

3.2.10 Custom error handling

--catch err=function

You may add custom error handling functions which catch script licensing errors. Error handler must 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 `sourceguardian` command:

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

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

Err tag	Returned code	Default message
ERR_LICENSE	01,02,03,04, 05	This script is not licensed to run on this machine. (code indicates a reason: 01-IP, 02-domain, 03-MAC address, 04-machine ID, 05-remote verification URL locking)
ERR_EXTLICCRC	06	A license file which is required to run this protected script is invalid...
ERR_EXPIRED	09	script has expired...
ERR_EXTLIC	13	script requires ... license file in order to run
ERR_OFFLINE	20	script requires the Internet connection in order to run
ERR_ALL	-	-

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

Custom error handler function should be defined before an error may occur. The best place for it is in the custom header code (see `-p` option) as it's loaded **before** any license checking is done and so error handlers will be always available if defined there. But you may also define a custom error handler function in another encoded file which is included before the script which may cause a license error. Don't put any passwords etc secret data to your custom header code as this code is stored unencoded within protected scripts, yet it is still protected by CRC against modifications.

Example:

Encode PHP script with a custom header containing definition of `my_err_handler()` function. Please note additional back slashes (\) are used to quote special character in the command line.

```
>sourceguardian -p "<?php function my_err_handler(\$code,\$msg) { echo sprintf('\nMy error handler caught error code %d with a message '%s'\n', \$code, \$msg); } ?>" --catch ERR_ALL=my_err_handler --external script.lic --projid Fg3161jd --projkey 826Gdb31 hello.php
```

The encoded script will have the defined code as unencoded in the beginning:

```
<?php ?><?php function my_err_handler($code,$msg) { echo sprintf("My error handler caught error code %d with a message '%s'\n", $code, $msg); } ?><?php
if(!function_exists('sg_load')){$_v=phpversion();$_x=explode('.',$_v);$_v2=$_x[0].'.'.(int)$_x[1];$_u=strtolower(substr(php_uname(),0,3));$_ts=(@constant('PHP_ZTS') || @constant('ZEND_THREAD_SAFE'))?'ts':'');$_f=$_f0='ixed.'.$_v2.$_ts.'.'.$_u;$_ff=$_ff0='ixed.'.$_v2.'.'.(int)$_x[2].$_ts.'.'.$_u;$_ed=@ini_get('extension_dir');$_e=$_e0=@realpath($_ed);$_dl=function_exists('dl') && function_exists('file_exists') && @ini_get('enable_dl') && !@ini_get('safe_mode');if($_dl && $_e && version_compare($_v,'5.2.5','<') && function_exists('getcwd') && function_exists('dirname')){$_d=$_d0=getcwd();if(@$_d[1]==':') {$_d=str_replace('\\','/',substr($_d,2));$_e=str_replace('\\','/',substr($_e,2));$_e.=( $_h=str_repeat('/..',substr_count($_e,'/')));$_f='/ixed/'.$_f0;$_ff='/ixed/'.$_ff0;while(!file_exists($_e.$_d.$_ff) && !file_exists($_e.$_d.$_f) && strlen($_d)>1){$_d=dirname($_d);}if(file_exists($_e.$_d.$_ff)) dl($_h.$_d.$_ff); else if(file_exists($_e.$_d.$_f)) dl($_h.$_d.$_f);}if(!function_exists('sg_load') && $_dl && $_e0){if(file_exists($_e0.'/'.$_ff0)) dl($_ff0); else if(file_exists($_e0.'/'.$_f0)) dl($_f0);}if(!function_exists('sg_load')){$_ixedurl='http://www.sourceguardian.com/loaders/download.php?php_v='.urlencode($_v).'&php_ts='.( $_ts?'1':'0'). '&php_is='.@constant('PHP_INT_SIZE'). '&os_s='.urlencode(php_uname('s')). '&os_r='.urlencode(php_uname('r')). '&os_m='.urlencode(php_uname('m')));$_sapi=php_sapi_name();if(!$_e0) $_e0=$_ed;if(function_exists('php_ini_loaded_file')) $_ini=php_ini_loaded_file(); else $_ini='php.ini';if((substr($_sapi,0,3)=='cgi')||($_sapi=='cli')||($_sapi=='embed')){$_msg="\nPHP script '$_FILE_' is protected by SourceGuardian and requires a SourceGuardian loader '$_f0.' to be installed.\n\n1) Download the required loader '$_f0.' from the SourceGuardian site:\n$_ixedurl.\n2) Install the loader to ";if(isset($_d0)){$_msg.=$_d0.DIRECTORY_SEPARATOR.'ixed';}else{$_msg.=$_e0;if(!$_dl){$_msg.="\n3) Edit '$_ini.' and add 'extension='$_f0.'" directive";}}$_msg.="\n\n";}else{$_msg.<html><body>PHP script '$_FILE_' is protected by <a href=\"http://www.sourceguardian.com/\">SourceGuardian</a> and requires a SourceGuardian loader '$_f0.' to be installed.<br><br>1) <a href=\"$_ixedurl.\">Click here</a> to download the required '$_f0.' loader from the SourceGuardian site<br>2) Install the loader to ";if(isset($_d0)){$_msg.=$_d0.DIRECTORY_SEPARATOR.'ixed';}else{$_msg.=$_e0;if(!$_dl){$_msg.="\n3) Edit '$_ini.' and add 'extension='$_f0.'" directive<br>4) Restart the web server";}}$_msg.="\n</body></html>";}die($_msg);exit();}}return sg_load('43C4F476BC939BBBAAQAAAAXAAAABKAAAACABAAAAAAAAD/GeeGIY7QwykhAfYlS+oi4pKKKDsEiCFXJTCu//P8cxRsVcNH0fVCa/3AF1LfTS/GNZ0ShuElC+Fl2LRdA5CGEGZKMYcdR+pRKIUkZeWyaggbhMYCDSKRJRNSRjyhKHJnnnjwv8m7fNADzAYGV80uanXpHGcQNmQ+eIZfyUENhcrCa+sZIH4rdSK9R8ucADlhF4CBgPfUmrrHqrUazGdkUkAAACgAAAA7p1Z1fxIoc1JvgcVNww2FXXTVUI8mo4SUSKtNz26hrdKWhet9sFKM/
```

```
gpKZJz2vQfQ6qXf9ElUPcdbX4lXfRTNyaxGYAUXBBg8L52N8//
BPwdX4QVi2FPhG9fCP8C9pvibWXLJq02OoqzLrzimVz0B2nlHqR2NCEoSgkbpvOD+N32CiOvqAzA/Zj9/
X+yksGsgK8oc70pqKr8JAA6zntCKQAAAAA=' );
```

Now a test run without the required license file:

```
>php -f hello.php
```

My error handler caught error code 13 with a message 'SourceGuardian Loader - This protected script requires script.lic license file in order to run. Please contact the author of the script to get a license file. Error code [13]'

Custom error handling with standard PHP error handling mechanism

You may catch some SourceGuardian errors using 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, strpos($errmsg, '[')+1,2);
        echo "SourceGuardian error $code"; // replace this with what you need for SourceGuardian errors
    } else
        echo $errmsg; // replace this with what you need for other PHP errors
    }
    error_reporting(E_ERROR);
    set_error_handler("myErrorHandler");
?>
```

3.2.11 Other options

There are some other options available to pass to sourceguardian command line encoder:

-v	Display version number
-h	Display full options list
--license	Display license information
--license--release	Release the current license which allows reinstalling to another machine or the same one. You get 3 free license resets for every license.
--credits	Display names of SourceGuardian developers
-w	Wait for key press before exit. Allows you to check encoding log in terminal before exit.

- q** Display settings and request confirmation. The encoder will display all encoding parameters and wait for a key press before any real encoding takes place. You may check all parameters and cancel if anything is not correct.
- verbose <n>** Controls the encoder output. 0-quiet, 1-print only errors to the log, 2-print standard log. 2 is default
- r{n}** Recurse subdirectories. The encoder will process all subdirectories recursively when searching files using specified file masks.
{n} is an optional directory trimming level, see below. IT IS IMPORTANT TO ENCLOSE FILE MASKS IN DOUBLE QUOTES. Otherwise, if file masks are not be enclosed in double quotes command line shell will expand file masks and recursion will not work as expected.
- b <ext>** 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!)
- x "mask" | @list** You may specify what files or directories shall NOT be encoded. You may specify either a strict name, relative path with a directory name or a mask (with ? and/or *).
- Example:
>sourceguardian -r -x "doc/*" -x "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
When specifying a relative path don't use ../ or ./ directory specifiers.
- f "mask" | @list** You may specify what files will be encoded by filenames, file masks or a file list. All other files which have been added for processing or found by expanding file masks 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:
>sourceguardian -r -f "*.php" -o "output_dir" ""
- All (with recursion) *.php files from the current directory will be copied and encoded 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 file masks with using of multiple -f

options:

Example 2:

```
>sourceguardian -r -f "*.php" -f "includes/*.php" -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.

You may enumerate all the files you want to encode in a file list. 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: -f @filelistname
When specifying a relative path don't use ../ or ./ directory specifiers.

-c "mask" | @list You may also use the -c (--copy) filter option in addition to -f (--file) and -x (--exclude). The -c (--copy) option may be used to specify what files will be copied as-is without encoding to the target folder. This option makes sense only if you specify the target folder with -o (--output) option. If -c is used without -o, then it works as -x and skips the specified files. The option may take * and ? wildcards or a @filelist.

-o <output_dir> You can specify an output directory for all encoded scripts. Source files will be unchanged if you specify an output directory different from your source scripts dir. The default backup option will be off when an output directory is specified. If you want to re-enable it, even when the output directory is specified, then use the -b <ext> option after the output directory option.
The full directory path to the source scripts will be recreated under the output directory if the full path to the source files was specified.

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

```
> sourceguardian -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.

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

Always quote file masks. Otherwise the command line shell will replace your mask with the real file and dir names and the result may be unexpected. You should always quote file masks that specify files to encode, copy or exclude in command line options (e.g. "*.php").

Recurring subdirectories

You may use wildcards in source directory names, e.g. /path/to/dir?/*.*.php This also works in @filelist and you may use it with -f, -t, -c, -x. If the @filelist is specified as source, recursion is automatically turned on, but it's still possible to change the directory trimming level with -r{n} if necessary.

Optional directory trimming

Optional directory trimming level may be specified with `-r{n}`. The default is 0 and means no trimming, this matches the mode used in previous versions of SourceGuardian. If `n` is specified, the encoder will remove `n` folder names from the beginning of file paths when encoding or copying the files to the target folder. This is similar to `-p` option of patch utility on Unix.

E.g. if you have the following directory structure in `/source`

```
/source/file0.php  
/source/dir1/file1.php  
/source/dir2/file21.php  
/source/dir2/file22.php
```

and encoding to the `/target` with the following command

```
sgencoder -o /target -r /source
```

encoding with default `-r` or `-r0` mode will create the following structure in the `/target` folder, i.e. the encoder recreates the full source path in the target

```
/target/source/file0.php  
/target/source/dir1/file1.php  
/target/source/dir2/file21.php  
/target/source/dir2/file22.php
```

Now you may use `-r{n}` if you don't need to recreate a full source path structure in the target

```
sgencoder -o /target -r1 /source
```

and get the following file structure in the target

```
/target/file0.php  
/target/dir1/file1.php  
/target/dir2/file21.php  
/target/dir2/file22.php
```

Now if you wonder why the default `-r` or `-r0` option may be useful, consider the following example

```
/project1/file0.php  
/project1/dir1/file1.php  
/project1/dir2/file21.php  
/project1/dir2/file22.php  
/project2/file3.php  
/project2/dir4/file41.php  
/project2/dir4/file42.php  
/project2/dir5/file5.php
```

Encoding with the following command in default mode works well

```
sgencoder -o /target -r /project1 /project2
```

```
/target/project1/file0.php  
/target/project1/dir1/file1.php  
/target/project1/dir2/file21.php  
/target/project1/dir2/file22.php  
/target/project2/file3.php  
/target/project2/dir4/file41.php  
/target/project2/dir4/file42.php  
/target/project2/dir5/file5.php
```

While encoding with trimming will create a mess of files from both projects which is obviously not what one would expect

```
sgencoder -o /target -r1 /project1 /project2
```

```
/target/file0.php  
/target/dir1/file1.php  
/target/dir2/file21.php  
/target/dir2/file22.php  
/target/file3.php  
/target/dir4/file41.php  
/target/dir4/file42.php  
/target/dir5/file5.php
```

So, you may use `-r{n}` when necessary, but the default mode with `n=0` is still useful, safe and always produce an expected result.

3.2.12 Encoding to standard output

SourceGuardian encoder may be used for encoding separate files taking source from standard input and sending encoded contents to standard output. In order to use the command line encoder in this mode, pass the `--` (double dash) instead of the input file name.

For example:

```
>sourceguardian -V5 -- < /path/to/source.php > /path/to/encoded.php
```

3.2.13 Exit codes

The encoder and command line tools may return the following exit codes. You may see the same codes displayed in brackets in the encoding log. When encoding a single file, the exit code may be used for checking if encoding was successful. When encoding multiple files and there are no issues with using the command line options, the encoder returns 0 (no error) and you need to check the encoding log to know further details.

Exit code	Reason
0	no error
1	file not found
2	php syntax or other compiler error
3	could not backup a file when backup is on

- 4 could not write output file
- 5 file is already encoded
- 6 license error
- 7 license error
- 8 usage error, check command line options
- 9 cancelled, no error but files were not encoded, e.g. help screen shown or license information
- 10 license expired (for trial version)
- 11 empty file, skipped
- 12 not a regular file, skipped
- 13 file copied without encoding
- 14 encoded in template mode
- 15 file skipped
- 18 multiple files are being encoded and an error happens during encoding (see encoding log for details)
- 255 other, internal or unexpected errors

3.3 Script license generator (full version)

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. You may find 'licgen' executable in the SourceGuardian installation directory in `/bin` subdirectory. Running the license generator without any options prints a list of all available options for a quick help. Please refer to this user manual for details of using the license generator.

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 13 and then create a license for each user with the SourceGuardian 13 Script License Generator.

Scripts encoded with the `--external` option 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 a protected script cannot find the specified license file it will return an error message: "SourceGuardian Loader - the script requires ... license file to run. Contact the script author regarding getting a license file. Error code [13]"

The algorithm used for locking scripts to an external license file 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 is the most powerful and flexible way to protect your scripts. We recommend that you use external license files for all your script protection.

A brief description of the algorithm

The algorithm uses an 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 this algorithm the encoder 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 "sourceguardian" 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:

```
>sourceguardian --external script.lic --projid "82Gi17Bn" --projkey "Az973Qq9" myscript.php
>licgen --projid "82Gi17Bn" --projkey "Az973Qq9" --days 7 script.lic
```

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

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

"SourceGuardian Loader - a license file required to run this protected script is invalid. Contact the script author regarding getting a license file. Error code [06]"

If a script is run with a license file with the 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 appears:

"SourceGuardian Loader - Loader - script checksum error. The encoded file has been modified. If the script requires a license file to run this error may be caused by invalid license file. Install original unmodified file or contact the script author regarding getting the original file or license file. Error code [12]"

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 if forgotten. They are required for generating licenses for your customers.

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

3.3.1 Usage

licgen [options] output.lic

--expire <dd/mm/yyyy>	Set license expiration date
--expire <00d[00h[00m[00s]]>	Set license expiration time from now
--domain <domain>	Bind license to domain name
--ip <x.x.x.x[/y.y.y.y]>	Bind license to ip/mask
--mac <x:x:x:x:x:x>	Bind license to mac address
--machine-id <machine id>	Bind license to a machine id
--projid <value>	Set project id (required, the same as for encoding)

--projkey <value>	Set project key (required, the same as for encoding)
--const name=value	Set custom defined constant
--time-server <server,server,...>	Set time server (for expiration date check)
--text "text"@file	Add plain text into the license file
-l	Display SourceGuardian license information for the tool itself
-w	Wait for key press before exit
-v	Display version number
-h	Display options help

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

It is possible to run the licgen tool with only a license file name but without any other locking options specified. It will generate the license required to run your scripts encoded with --external option but no locking will be applied to the protected scripts. To enable locking with the external license file please read about [script locking options](#). All locking options work the same as similar options of the sourceguardian executable.

You may use a -- (double dash) instead of the output file name in order to send licgen's output to console instead of a file which may be useful for automating license generation when running licgen on the server side.

Locking options

Most of the options work exactly the same as options of the encoder. Please refer to the [Script locking options](#) section for further details. Note, some of the options are available only for the encoder and not the license generator, e.g. --ip-encrypt, --domain-encrypt, --machine-encrypt, --remote-verification-url and some others.

Custom constants

You may add custom constants to license files exactly the same as you do it with the encoder. Please refer to the [Custom predefined constants](#) section for further details.

Options unique to the license generator

--text "text"

This option lets you add a custom text that will be embedded as-is into the license file and therefore that text is readable. The text is protected with a checksum against modification. You may include any text such as user information, license description etc.

All user {constants} that are defined with the --const option above will be replaced in the text. Also some standard SourceGuardian constants may be used:

- {SG_DATE} - current date i.e. date of encoding
- {SG_LICENSEE} - SourceGuardian license owner from the SourceGuardian license file (i.e. your name, not your customer's name)

{RG_EXPIRY_DATE} - your custom license expiry date. Changing of this text in the license file does not make sense and is NOT opening a back door, if you use the expiry date in your custom licenses. The expiry date is stored encrypted as well as other options within the license file. The readable text is only for information and you may put a name of your product, name of your customer and the expiry date.

Constant names are **case sensitive**. It works in the same way also for the custom header in protected scripts. [See details](#)

The text contents may be loaded from a file: --text @textfile. All the constants replacements will be done in that case too.

3.4 File information tool (full version)

The Information Tool is an external tool for viewing protected scripts and script license files details. You may find the 'sginfo' executable in the SourceGuardian installation directory in /bin subdirectory. Running the information tool without any options prints a list of all available options for a quick help. Please refer to this user manual for details on using the 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 encode date, expiration date, locking options etc that were used during file encoding or script license generation. You may pass the encoded script name or script license as a parameter to this tool.

Script information: sginfo [options] file.php

License information: sginfo [options] file.lic

You might need to specify the 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.

3.4.1 Usage

Script information: sginfo [options] file.php

Options:

- | | |
|------------------------------|--|
| --tag-ip [x.x.x.x/{y.y.y.y}] | Target IP for decrypting. Required to view information about scripts encoded with --ip-encrypt option. |
| --tag-domain [domain] | Target Domain for decrypting. Required to view information about scripts encoded with --domain-encrypt option. |
| --projkey [value] | Script Project Key for decrypting. Required to view full information about scripts protected with an external license file when such a file is not available. You need to specify the project key to decrypt the bytecode section and view information about the supported PHP versions and test |

the integrity of the bytecode.

If the protected script is locked to an external license file and this file is also available in the script's directory or parent directories then all information extracted from this external license file will be also automatically displayed.

License information: rginfo [options] file.lic

Options:

--projid [value] License Project ID for decrypting an external license file. Required to view information about external license file generated with SourceGuardian License Generator.

Other options:

-v	Display version number
-h	Display full options list
-l	Display license information

SourceGuardian 13 for macOS, Windows, Linux

Part



IV

4 Running protected scripts

Scripts protected with SourceGuardian™ require the installation of a SourceGuardian™ loader on the target machine in order to run. Protected script loaders are dynamically loaded PHP extensions which load the protected script, decrypt it and then run the bytecode. The source code is never restored at any time, even in memory. There are different versions of the loaders available for different operating systems and PHP versions. Protected scripts will automatically attempt to find the loader in the `ixed/` subdirectory located within the protected script's directory or parent directories. SourceGuardian™ loaders may also be installed into PHP's `extension_dir` folder and the `php.ini` configuration file. This is the only way to run protected files if automatic loading is not supported by your OS and PHP or if faster performance is required. We recommend that you install the loader to PHP's `extension_dir` and `php.ini` even if dynamic automatic loading is possible.

For PHP versions 5.2.5+, SourceGuardian™ loaders must be installed into the PHP extensions directory (`extension_dir`). You may find the `extension_dir` path in the `php.ini` configuration file or in the `phpinfo()` output. A 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+.

We periodically update SourceGuardian™ Loaders. The latest loaders are always freely available from <http://www.sourceguardian.com/loaders/>

4.1 Installing loaders

SourceGuardian loader should be installed into the `PHP extension_dir` and `php.ini` for PHP 5.2.5+ due to limits in PHP engine. The reason is that SourceGuardian loader is a PHP extension and it's true for any other PHP extension for PHP since 5.2.5.

We recommend that you use a built-in loaders helper from File/Install Loaders menu. Please read [this section](#) in the user manual to know how to use this option.

Alternatively you may use our online loader assistant to know the loader you need and how to install it to your target system. The loader assistant is always available from our web site <http://sourceguardian.com/loaders/download.php>

If you still want to install the loader manually please follow instructions below.

Although it's possible to use automatic loading for old PHP versions in some conditions, if you use PHP < 5.2.5 you still need to install the loader manually if:

1) Operating system and PHP mode:

Linux, FreeBSD, OpenBSD, MacOSX, other UNIX - PHP installed as a webserver's module (with thread safety on)

Windows - PHP installed as webserver's 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 always has Thread Safety on.

- 3) If dl() is disabled. You have "enable_dl=Off" setting in the php.ini configuration file.
- 4) If safe_mode is on.

Usually 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. Having SourceGuardian™ loader installed to extension_dir and php.ini you give the maximum performance for your protected scripts. A reason is the script does not need to search for the loader every time it runs.

To install a SourceGuardian™ loader 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 /Loaders subdirectory within SourceGuardian main installation directory and copy it to the PHP extension directory (extension_dir - check the phpinfo() output). We also suggest that you check our site for an updated version of loaders <http://www.sourceguardian.com/loaders/>**
- 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). This will depend on your OS, PHP version, Thread Safety mode. Please refer to the "Loader filename structure" section below.**
- 4) Restart the webserver in order to apply changes done in the php.ini configuration file and reload PHP.**
- 5) Optionally you may open the phpinfo() page now to check that SourceGuardian loader has been successfully installed - search for "SourceGuardian" string.

Important information for Windows as a target platform for running protected files.

- a) If PHP engine is installed as a dynamic library and loaded automatically during web server start then the PHP engine is probably compiled with thread safety ON. It means that you need to use "ts" version of the loader. If you use all-in-one package like WAMP or XAMPP then the included PHP engine is also compiled with thread safety ON and you need to use "ts" version of the loader in that case, e.g. ixed.5.3ts.win for PHP 5.3.x.
- b) There are VC6 and VC9 version of loaders for Windows. VC6 versions are compiled with the legacy Visual Studio 6 compiler. VC9 versions are compiled with the Visual Studio 2008 compiler. If you are using PHP with Apache1 or Apache2 from apache.org you need to use the VC6 versions of PHP and the loader. If you are using PHP with IIS or third-party builds of Apache you should use the VC9 versions of PHP and the loader. Usually you do not use VC9 version with apache.org binaries.
- c) There are loaders for 64-bit version of Windows for PHP 5.3+ VC9. Earlier versions of PHP have no officially supported sources for 64-bit Windows and cannot be compiled in this environment. As a result SourceGuardian has support for Windows 64-bit loaders for PHP 5.3+. Only VC9 versions are supported for 64-bit Windows.

Examples of SourceGuardian loader names:

extension=ixed.4.3.lin	# for Linux, non thread safe, PHP 4.3.x
extension=ixed.5.0.0.lin	# for Linux, non thread safe, PHP 5.0.0
extension=ixed.5.0.1.lin	# for Linux, non thread safe, PHP 5.0.1
extension=ixed.5.0.2.lin	# for Linux, non thread safe, PHP 5.0.2
extension=ixed.5.0.lin	# for Linux, non thread safe, PHP 5.0.3+
extension=ixed.5.1.lin	# for Linux, non thread safe, PHP 5.1.x
extension=ixed.5.2.lin	# for Linux, non thread safe, PHP 5.2.x
extension=ixed.5.3.lin	# for Linux, non thread safe, PHP 5.3.x
extension=ixed.5.0ts.lin	# for Linux, thread safe, PHP 5.0.3+
extension=ixed.4.3.fre	# for FreeBSD, non thread safe, PHP 4.3.x
extension=ixed.4.3ts.win	# for Windows, thread safe, PHP 4.3.x
extension=ixed.5.2ts.win	# for Windows, thread safe, PHP 5.2.x
extension=ixed.5.3ts.win	# for Windows, thread safe, PHP 5.3.x

4.2 Automatic loading

If you use PHP < 5.2.5 there is a chance you can use automatic loading. It means that protected scripts will be able to find an appropriate loader automatically and use it. However, if you have permissions to install the loader server-wide to extension_dir and php.ini we strongly recommend that you do it. Server-wide installation of the loader increases performance of the protected scripts.

A protected script will be able to find and load an appropriate loader if:

1) Operating system and PHP mode:

Linux, FreeBSD, OpenBSD, MacOSX, other UNIX - PHP is installed as CGI or CLI

Linux, FreeBSD, OpenBSD, MacOSX, other UNIX - PHP is installed as a webserver's module (with thread safety off)

Windows - PHP is 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 loaders are installed to the ixed/ subdirectory within your scripts directory or any parent directory.

6) **PHP version is older than 5.2.5.**

7) safe_mode is off.

8) (For Windows) extension_dir= option in php.ini should point to the directory located **on the same drive** with your document root and scripts directory.

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 to 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.

Example 1:

(loaders are in the `/ixed/` subdirectory within the `scripts` directory)

<code>/home/mysite/www/myscript1.php</code>	- your protected script(s)
<code>/home/mysite/www/myscript2.php</code>	- your protected script(s)
<code>/home/mysite/www/subdir/otherscript1.php</code>	- other protected script(s)
<code>/home/mysite/www/subdir/otherscript2.php</code>	- other protected script(s)
<code>/home/mysite/www/ixed/ixed.*</code>	- SourceGuardian loaders

Example 2:

(loaders are in the `/ixed/` subdirectory within a parent directory)

<code>/home/mysite/www/myscript1.php</code>	- your protected script(s)
<code>/home/mysite/www/myscript2.php</code>	- your protected script(s)
<code>/home/mysite/www/subdir/otherscript1.php</code>	- other protected script(s)
<code>/home/mysite/www/subdir/otherscript2.php</code>	- other protected script(s)
<code>/home/ixed/ixed.*</code>	- SourceGuardian loaders

4.3 Loader filename structure

The following provides an overview of the SourceGuardian 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)
- Z** - minor PHP version number (2 for 5.0.2)
This part may be missed in the loader name which means that this loader is for all higher PHP versions, e.g.:
 - `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.x
 - `ixed.5.2.lin` - for all PHP 5.2.x
 - `ixed.5.3.lin` - for all PHP 5.3.x
- dd** - optional code
 - (if missed) - this is the loader for non-thread safe version of PHP. Most UNIX installations are non-thread safe.
 - `ts` - this is the loader for thread safe version of PHP. Most Windows installations are thread safe, most Unix installations are not.
- os** - three char code of operating system type.
 - `.win` - Microsoft Windows
 - `.lin` - Linux (32 and 64 bit versions available)
 - `.fre` - FreeBSD (32 and 64 bit versions available)

- .ope - OpenBSD (32 and 64 bit versions available)
- .dar - OS X (universal binary version)

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 at the PHP level. However, 32-bit and 64-bit loaders are packed in different zip (tar.gz, tar.bz2) files so you can easily distinguish them. You need to use a correct 32-bit or 64-bit version of the loader for your system according to the platform and options 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 a 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 if your PHP is 32-bit or 64-bit using the command line "file" tool, e.g. "file /path/to/php".

Note, some operating systems such as Windows or Linux may no problem run 32-bit executable including PHP even if the OS itself is 64-bit. So, you may have 32-bit PHP installed and running on a 64-bit OS. This may cause misunderstanding, please check if your PHP is 32 bit or 64 bit, not the OS. However, if your are running 32-bit OS then PHP can definitely be only 32-bit too.

4.4 Zend extension support

SourceGuardian loaders may be loaded as Zend extensions. This lets you specify a 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 the loader for non thread-safe PHP use `zend_extension` option in `php.ini`:

```
zend_extension = /usr/local/ixed/ixed.7.3.lin
```

For thread-safe PHP use `zend_extension_ts` option in `php.ini`: (mod_php apache module is always thread safe in Windows)

```
zend_extension_ts = /usr/local/ixed/ixed.7.3ts.lin
```

You need to specify an appropriate loader for your OS and PHP version. See [loaders filename structure](#) section.

4.5 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 must be set in the `php.ini`:

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

Note: The files must be encoded with "conjunction" option enabled in order to run with the above setting enabled. Enable "Script will only work with other encoded files" in [Locking settings](#) or use `--conj` option for CLI encoder.

If any unencoded script is executed one of the following error message appears:

Error code [08] text changed to:

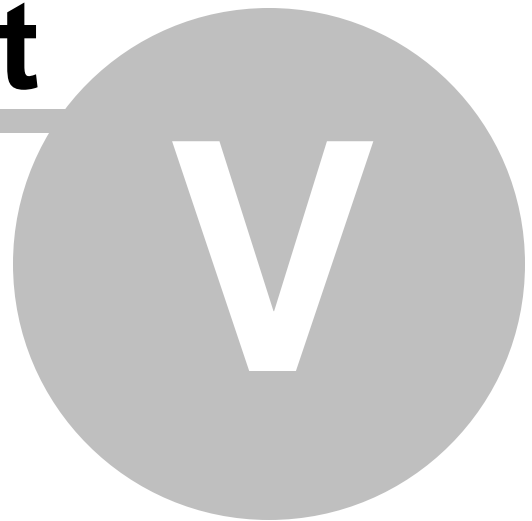
PHP Fatal error: SourceGuardian Loader - This script ... or other unencoded scripts cannot run in this environment. Please contact the author of the script regarding this problem. [08]

Error code [15] text changed to:

PHP Fatal error: SourceGuardian Loader - This protected script ... can run only in conjunction with other encoded files of the same project. Please contact the author of the script regarding this problem. Error code [15]

SourceGuardian 13 for macOS, Windows, Linux

Part



5 Encoding of HTML templates and other non-PHP files

You may encode HTML templates, custom configuration files or other non-PHP files using the SourceGuardian encoder. HTML templates or other non-PHP files may be encoded by the encoder and then read and decrypted from the protected PHP code using SourceGuardian API function `sg_load_file()`. Please refer to the [section below](#) to know how you can use encoded templates from your encoded PHP files. Files encoded in this mode cannot be automatically run by the PHP engine.

If you use HTML page templates that PHP engine runs directly, encode them in normal PHP mode, you do not need to encode them as non-PHP.

Encoded templates and other non-PHP files look like this:

`SourceGuardianAAwAAAAFCgAAAAZ0jwEA/9QAMUp+g+GpvG3vbvYj4Is=`

Within this document we will refer to HTML templates or other non-PHP files simply as "templates" for short. There is no difference for the encoder between HTML templates, other templates or any other non-PHP files.

Template files 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 as a different SourceGuardian project.

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

SourceGuardian GUI generates `project_id` and `project_key` automatically for a new project. `Project_id` and `project_key` are saved within your SourceGuardian project file. Please always use the same project for adding/changing encoded templates otherwise old templates cannot be used with newly encoded scripts or vice versa because of new `project_id` and `project_key`. Save your `project_id` and `project_key` values for future use. The `project_key` value is also required for correct license generation if you use locking to a license file.

5.1 GUI

Encoding templates and other non-PHP files is simple in SourceGuardian GUI. In the [project window](#) choose "Custom non-PHP" type for files or folders that you would like to encode as non-PHP. These files will be encoded when you click Encode.

5.2 Command line interface

Use the `-t` option to specify files, file masks or file list for your custom non-PHP files.

Example 1:

```
>sgencoder -r -t"myproject/templates/*.tpl" "myproject/"
```


You may specify multiple `-t` options if you need. All other files which are not specified as non-PHP 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 non-PHP and all other files will not be encoded and will be copied to the output directory as-is. Output directory may be specified by the `-o` option.

You may use file lists for specifying your custom non-PHP files and use file masks as well as normal file names in the list.

Example 2:

if 'mytemplates' file contains:

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

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

This command will encode files specified in 'mytemplates' (i.e. templates/mytemplate.txt, *.tpl, *.html and *.htm files in source_dir) as templates, it will encode all other files in source_dir as PHP scripts.

Example 3:

if additionally 'myphpfiles' file contains:

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

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

This command will encode *.tpl, *.html and *.htm files in source_dir as templates, *.php and *.inc files as PHP scripts and will leave all other files from source_dir unencoded but copied to the output_dir. See details below about `-f` option.

5.3 Using encoded non-PHP files

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

```
$template_data = sg_load_file($filename);
```

`sg_load_file()` may generate the following errors:

SourceGuardian Loader - Encoded template file ... is not found. Contact the script author about this problem. Error code [21]
(when the loader could not find a specified template file)

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

SourceGuardian Loader - Decryption error for encoded template file ... Install the original unmodified file or contact the script author to get the original file. Error code [23]
(an error has been detected at the decoding stage, possibly because the template that you are trying to

load was encoded for another SourceGuardian project - different project_id or project_key)

SourceGuardian Loader - Error loading encoded template file ... Check file permissions or contact the script author about this problem. Error code [24]
(system error when loading a template file - insufficient memory, read error etc)

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

5.4 Using encoded 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 3.0 but it should be easy to update other versions too. Please read 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 Smarty.class.php function fetch() and function _smarty_include()

replace:

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

with:

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

In internals/cordite_file.php function smarty_core_write_file()

replace:

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

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

with:

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

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

After all the above changes are done the Smarty engine can work with normal unencoded templates when runs from unprotected scripts and encoded templates when runs from SourceGuardian encoded scripts. 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.

5.5 Creating custom encoded files from protected scripts

If your script generates files online and you need to secure them, it's possible with SourceGuardian. You may use `sg_encode_file($filename, $data)` SourceGuardian API function for encoding a file from protected code. This file will be encrypted in the same way as the SourceGuardian encoder encodes template files.

```
sg_encode_file($filename, $data);
```

Important security notice. A built-in SourceGuardian API encoder (`sg_encode_file()` API

function) is suited only for encoding templates, configuration, 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 ... Check file permissions or contact the script author about this problem. Error code [25]

(The 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 a 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 get encrypted using the current protected script's project identifier (`project_id`) and key (`project_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. These protected templates can be decrypted only on the machine with same IP (or domain name).

Since SourceGuardian version 8.1 we have changed the way how script keys based on Project ID or Project Key (if an external license file is used) are used for encoding or decoding files with `sg_load_file()` and `sg_encode_file()` loader API functions. Now we keep track of what protected PHP script uses which key and the loader will use an appropriate key for encryption/decryption user files and strings (see new `sg_encode_string()` and `sg_decode_string()` API functions). Older versions of the loader would use the key of the last loaded protected script. These changes allow to use the encoding functions correctly in a situation when one protected script is inherited from another protected script and the encoding/decoding functions are called.

5.6 Using encoding SourceGuardian API from unprotected script

SourceGuardian API functions are part of the SourceGuardian loader and they 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 returns 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 writes the file's data as-is without encryption:

- when runs from an unprotected script with the loader installed server-wide

5.7 Debugging of scripts which work with encoded templates

Usually SourceGuardian API functions 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 that. For convenience and easy debugging we suggest two possible ways 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 a development machine as normally PHP installation is over developer's control there. If the loader is installed server-wide, 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 completed and project is encoded, SourceGuardian API functions will start working in normal (protected) mode for reading and writing encoded templates/non-PHP files data.

2) Use our SourceGuardian API stub script `sgapistub.php` (<http://sourceguardian.com/knowledgebase/sgapistub.zip>) 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 the protected script and SourceGuardian API functions are available this script does nothing and lets real API functions work. To use this stub script you need to include it from your script that 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 debug the script and check content of the output file or loaded template file. When project debugging is completed 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 directive 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.

SourceGuardian 13 for macOS, Windows, Linux

Part



VI

6 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.

6.1 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 when running them. While normally you will get an error message from SourceGuardian loader saying about a CRC error if running the modified protected script, running modified protected scripts may cause serious problems like segmentation faults in the PHP interpreter. This is not a problem with SourceGuardian or loaders.

6.2 Error messages during encoding

You will see the log file printed in the terminal window during an encoding process. Encoding status message will be displayed for each encoded file. You may get the following status messages:

ok	The file was encoded without problem.
file not found cannot be read	The specified file could not be found. Check the specified file path.
PHP syntax or other compiler error	The original file has syntax or other errors and thus cannot be encoded. Check your file, test it with the PHP interpreter. This error usually appear when encoding for multiple versions of PHP and if your file is not compatible with all the target versions of PHP. Make sure your PHP script is compatible with all versions of PHP you are encoding for.
could not backup source file, skipped	The encoder could not make a backup copy of your original file (when no output directory was specified). SourceGuardian skips the file in that case to keep your original version. Check you have enough free space available and permissions to write to your original files directory.
cannot not write file	The encoder could not write the encoded file. Check you have enough free space available and permissions to write to original files directory or to the output directory if you have specified it with the -o option.
file is already processed by SourceGuardian	The encoder will not encode files which are already encoded with SourceGuardian. Check your original files directory.
empty file, skipped	The encoder will not encode empty files. If you need to have empty files for any reasons you may copy them manually.
not regular file, skipped	The encoder could not encode a file because it is not a regular file. It may be a socket or a unix device for example.
do not encode, skipped	The file was marked as 'do not encode' and therefore was skipped.
copied	The file was copied without encoding. It is possible when -f option is used to specify files to encode and -o option is used to specify output directory. All other files than specified in -f option will be copied as-is without encoding. It is useful for encoding an entire project directory when it also may contain non-PHP files.

internal encoder error,
unknown error

This is an internal problem with the encoder. Check you have enough free memory space to run the encoder and some free space on the disk. If it is not a memory problem then let us know about this error. Send an email to support@sourceguardian.com with a detailed description of the error and the command line used for running the encoder. We will investigate the problem. We may also need some additional information from you.

6.3 Error messages when running protected scripts

This section includes descriptions of the error messages that may appear when running protected scripts. They are not in any particular order, but we would suggest that you take a look at this section before you contact us regarding any support matter.

PHP script '...' is protected by SourceGuardian and requires a SourceGuardian loader ... to be installed.

An appropriate SourceGuardian Loader is not installed. To run protected scripts on the target machine you need to install the SourceGuardian Loader. We suggest that you use the loader assistant to know the loader you need and how to install it: <http://sourceguardian.com/loaders/download.php> or use a built-in [loader installation tool](#).

SourceGuardian Loader - This script is not licensed to run on this machine or it is running out of web server environment. Run this script in web server environment. Please contact the script author about this problem. Error code [01]

The protected script was encoded with the --ip locking option and was bound to some IP address(es). When the script is run on the web server from the other IP address the above error message appears. Check the IP address of the machine running the script. Check the script is running in a web server environment and the server IP address is available as an environment variable for the script. See [script locking options section](#) in this manual for details.

SourceGuardian Loader - This script is not licensed to run on this machine or it is running out of web server environment. Run this script in web server environment. Please contact the script author about this problem. Error code [02]

The protected script was encoded with a --domain locking option and was bound to some domain name(s). When the script is about run on the web server from the other domain the above error message appears. Check the domain name in the URL accessing the script. Check the script is running in a web server environment and that the server domain name is available as an environment variable for the script. See [script locking options section](#) in this manual for details.

SourceGuardian Loader - This script is not licensed to run on this machine. Please contact the script author about this problem. Error code [03]

The protected script was encoded with a --mac locking option and was bound to some LAN hardware MAC address(es). When the script is about run on a machine running a different MAC address the above error message appears - MAC address(es) specified for the script during encoding do not match any of

MAC addresses of the target machine. Check the MAC address(es) on the machine running the script. We suggest to use "ifconfig -a" command for it. Execute it in from the command line on the target machine to list all the available network interfaces. See [script locking options section](#) in this manual for details.

SourceGuardian Loader - This script is not licensed to run on this machine. Please contact the script author about this problem. Error code [04]

The same as above but the problem is related to [machine ID locking](#) and running the script from the machine which is not allowed.

SourceGuardian Loader - This script is not licensed to run on this machine. Please contact the script author about this problem. Error code [05]

The same as above but the problem is related to [remote verification](#) URL locking for CLI scripts.

SourceGuardian Loader - A license file required to run this protected script is invalid. Contact the script author for getting a license file. Error code [06]

The protected script was encoded with an --external locking option and is bound to the external license file. That license file is required to run the protected script. The license file has been found but it failed validation. Possibly it was accidentally changed. Install the correct license file for this project or generate a new license file with the same Project ID and Project Key values that were used for encoding the script. Refer to the [script locking options](#) section in this user manual for details.

SourceGuardian Loader - This protected script does not support version ... of PHP. Please contact the author of the script about this problem. Error code [07]

SourceGuardian supports encoding for multiple versions of PHP. You need to use the --phpversion option for this in the SourceGuardian command. The above message says that the script was not encoded for the version of PHP currently running the script. This may happen, for example if you encode the script for PHP 7.2 (--phpversion 7.2 option) and try to run it under PHP 7.3. Make sure the script is encoded for the version of PHP running on your target server and that it is compatible with that version of PHP (otherwise you will get an error message during encoding). Please note, if you do not specify --phpversion option in SourceGuardian command, your scripts will be encoded for all the supported versions of PHP.

SourceGuardian Loader - This script has expired. Please contact the script author about this problem. Error code [09]

The protected script was encoded with an expiration date set for the script or within the external license file. The above message says that the period of using the script has expired and you cannot use this script anymore.

SourceGuardian Loader - Script ... header is broken. The encoded file has been modified. Install the original unmodified file or contact the script author for getting the original file. Error code [10]

SourceGuardian Loader - Script ... is broken. The encoded file has been modified. Install the original unmodified file or contact the script author for getting the original file. Error code [11]

SourceGuardian Loader - Script ... loader checksum error. The encoded file has been modified. Install the original unmodified file or contact the script author for getting the original file. Error code [17]

SourceGuardian Loader - Decompression error status The encoded file has been modified. Install the original unmodified file or contact the script author for getting the original file. Error code [18]

The SourceGuardian Loader performs a number of validation actions for the protected script before it runs the script. The above messages says that some validation has failed. It may happen because the script has been changed, possibly accidentally. Any changes to the protected script are restricted for security reasons. It is not allowed to make changes to the protected script. Do not change the protected script, install original file or re-encode the script again. Unprotected areas of the script (shell command, loader code, your custom header code or custom error code) are still protected with CRC and should not be changed either. Different messages above indicate different validation stages that the loader performs before running the protected script.

If you need to change or update a protected PHP script, change it and re-encode.

SourceGuardian Loader - Script ... checksum error. The encoded file has been modified. If this script requires a license file to run this error may be caused by an invalid license file. Install the original unmodified file or contact the script author for getting the original file or license file. Error code [12]

This error message may be caused by the same reasons as the error messages above. Additionally this error may happen when the protected script had been locked to an external license file but the wrong license file was installed on the target machine. I.e. the license file was found by name but it is incorrect, generated for a different project or is broken. Therefore decryption of the protected script bytecode has failed. It is important to use the same Project ID and Project Key values for the license file that were used for encoding the script. Refer to the [script locking options](#) section in this user manual for details. Do not change the protected script or a license file - install the original file or re-encode the script again. If an external license file is used - install correct license file for this project or generate a new license file with the same Project ID and Project Key values that were used for encoding the script.

SourceGuardian Loader - This script requires ... license file to run. Contact the script author for getting a license file. Error code [13]

Protected script was encoded with --external locking option and is bound to an external license file. That license file is required to run the protected script on the target machine. Name of the license file should match the name specified during encoding. Use SourceGuardian License Generator to create a license file and Install it on a target machine into the script's directory or any parent directory. If the license file is already there - check if the script has enough permissions to read the license file. It is important to use the same Project ID and Project Key values for the license file that were used for encoding the script. Refer to [using scriptlicense generator](#) section in this user manual for details.

SourceGuardian Loader - Evaluation loader has expired. This file has been encoded with evaluation version of SourceGuardian. Please download and install [http://www.SourceGuardian.com/loaders/ latest loaders](http://www.SourceGuardian.com/loaders/latest loaders). Error code [14]

The SourceGuardian Loader you installed on a target machine is expired. This may happen only with the scripts encoded by the demo version of SourceGuardian.

Download and install updated loaders from <http://www.SourceGuardian.com/loaders/>

SourceGuardian Loader - Incompatible loader version. The file has been encoded with a newer version of SourceGuardian. Please download and install [http://www.SourceGuardian.com/loaders/ latest loaders](http://www.SourceGuardian.com/loaders/latest loaders). Error code [19]

Your protected script was encoded with a newer version of SourceGuardian than the installed loader.

Download and install updated loaders from <http://www.SourceGuardian.com/loaders/>

SourceGuardian Loader - This script requires an internet connection to run. The file has been encoded to run only when an internet connection is available. Setup an internet connection. Error code [20]

The protected script was encoded with --time-server option or this option was specified for the external license file the script is bound to. The SourceGuardian Loader is trying to connect to the specified time server(s) using NTP and TIME protocols. If it is not possible to connect, for any reason (connection lost, error etc) the above error message will appear. Check that the time servers specified for the protected script or script license file exist. Use the SourceGuardian Information tool to know what time servers (or any other options) were specified during encoding. We suggest that you specify a number of available time servers so your script can run even if some time servers are temporary offline. See the [script locking options](#) section in this user manual for details.

SourceGuardian Loader - Insufficient memory. Error code [FF]

This may happen if the SourceGuardian Loader failed to allocate some memory (RAM) using standard library functions. Check your system has enough free memory for running the protected script.

SourceGuardian Loader does not need much free memory available, so this error usually never appears. Otherwise your system cannot work anyway without free memory available.

SourceGuardian Loader - Internal error: ...

Something wrong happens during the protected script execution. If you see this error please contact support to let us know about it support@sourceguardian.com.

Based on our experience in supporting SourceGuardian we must say that most protected scripts errors are caused by three factors: absence of loaders, modifications in the protected script or a script license or locking options used during encoding. We suggest that you use SourceGuardian Information Tool to know details about locking options for your scripts and script license files. See [using information tool](#) section in this user manual for details.

6.4 Extension directory (php.ini setting)

If you want the 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 must point to a directory that does exist on the server. If it doesn't exist then PHP cannot load any extension at all (including the 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.

Note, dynamic loading of the loaders is available only for PHP < 5.2.5

6.5 Getting "This protected script can run only in conjunction..." error

Getting "SourceGuardian Loader - This protected script can run only in conjunction with other encoded files of the same project" while using a correct license file and no other non-encoded files are prepended or included from the protected one?

Check that `xdebug` or another PHP debugging extension is off in the `php.ini`. Such extensions modify the bytecode on-the-fly and this makes the SourceGuardian loader think that the protected script has been modified or non-SourceGuardian encoded code is running.

SourceGuardian 13 for macOS, Windows, Linux

Part



VII

7 Advanced users

7.1 PHP shell scripts encoding

SourceGuardian supports encoding of PHP scripts which are UNIX shell scripts. SourceGuardian keeps the first line unchanged for the script if it begins with a `#!` UNIX shell script prefix (ex. `#!/usr/bin/php`) This lets encoded 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. (This also means that the path that may be specified in the first line cannot be changed after the file has been encoded).

7.2 SourceGuardian loader API

SourceGuardian loader defines some functions which are **available from protected scripts**. These functions are available only when the loader is loaded into PHP engine and protected script is running.

array sg_get_mac_addresses()

This function returns an array of hardware addresses (MAC-addresses) of all network interfaces installed on the machine where the script is running. It may be useful for creating custom locking schemas etc. Each MAC address is returned as formatted string that includes 6-byte hardware address in the following format: `XX:XX:XX:XX:XX:XX` Up to 32 network hardware addresses may be returned.

string sg_get_const(\$name)

Returns the value of a custom constant that was during encoding or a value of the predefined constant. See [Custom predefined constants](#) section for details.

string sg_load_file(\$filename)

Loads and decrypts the encoded template and returns contents as a string. See [Using protected templates](#) section for details.

sg_encode_file(\$filename, \$data)

Encodes template or another file from the protected code. See [Creating custom encoded files](#) section for details.

string sg_encode_string(\$data)

Encrypts a string. An internal key based on Project ID or Project Key (if the script is bound to an external license file) is used for encryption. An encoded string is converted to base64 and can be easily saved to a text file. This function and `sg_decode_string()` function are useful for storing critical user settings in encrypted format.

Note: if this function is called from unencoded code it will return unmodified string. This lets you debug your scripts but the actual encoding will work only when the function is called from the protected code.

string sg_decode_string(\$encrypted)

Decrypts a string previously encrypted with `sg_encode_string()`. An internal key based on Project ID or Project Key (if the script is bound to an external license file) is used for decryption. This function and `sg_encode_string()` functions are useful for storing critical user settings in encrypted format.

Note: if this function is called from unencoded code it will return unmodified string. This lets you debug your scripts but the actual decoding will work only when the function is called from the protected code.

7.3 Marking a file to be skipped during encoding

It's possible to mark a file to be skipped by the encoder. Add the following string anywhere in the code, use comments. Skipped files will be copied as-is to the target folder if it's specified.

SourceGuardian:DO_NOT_ENCODE

E.g. `/* SourceGuardian:DO_NOT_ENCODE */`

Note: Comparison is case sensitive for Windows. Do not change or mix the case for better code compatibility.

SourceGuardian 13 for macOS, Windows, Linux

Part



8 License

8.1 SourceGuardian License

PLEASE READ THIS CAREFULLY BEFORE USING MATERIALS

A IMPORTANT PROVISIONS

YOUR ATTENTION IS DRAWN PARTICULARLY TO THE PROVISIONS OF CLAUSE 10.2 OF THE FOLLOWING LICENCE AGREEMENT.

B PROPERTY OF SOURCEGUARDIAN

YOU MAY OBTAIN A COPY OF THIS SOFTWARE PRODUCT EITHER BY DOWNLOADING IT REMOTELY FROM OUR SERVER OR BY COPYING IT FROM AN AUTHORISED DISKETTE, CD-ROM OR OTHER MEDIA ('HARD MEDIA'). THE COPYRIGHT, DATABASE RIGHTS AND ANY OTHER INTELLECTUAL PROPERTY RIGHTS IN THE PROGRAMS AND DATA WHICH CONSTITUTE THIS SOURCEGUARDIAN (VERSION 10) SOFTWARE PRODUCT (THE 'MATERIALS'), TOGETHER WITH THE HARD MEDIA ON WHICH THEY WERE SUPPLIED TO YOU, ARE AND REMAIN THE PROPERTY OF SOURCEGUARDIAN LIMITED ('SOURCEGUARDIAN'). YOU ARE LICENSED TO USE THEM ONLY IF YOU ACCEPT ALL THE TERMS AND CONDITIONS SET OUT BELOW.

C LICENCE ACCEPTANCE PROCEDURE

BY CLICKING ON THE ACCEPTANCE BUTTON WHICH FOLLOWS THIS LICENCE AGREEMENT (MARKED 'I ACCEPT'), YOU INDICATE ACCEPTANCE OF THIS LICENCE AGREEMENT AND THE LIMITED WARRANTY AND LIMITATION OF LIABILITY SET OUT IN THIS LICENCE AGREEMENT. SUCH ACCEPTANCE IS EITHER ON YOUR OWN BEHALF OR ON BEHALF OF ANY CORPORATE ENTITY WHICH EMPLOYS YOU OR WHICH YOU REPRESENT ('CORPORATE LICENSEE'). IN THIS LICENCE AGREEMENT, 'YOU' INCLUDES BOTH THE READER AND ANY CORPORATE LICENSEE.

D LICENCE REJECTION PROCEDURE

YOU SHOULD THEREFORE READ THIS LICENCE AGREEMENT CAREFULLY BEFORE CLICKING ON THE ACCEPTANCE BUTTON. IF YOU DO NOT ACCEPT THESE TERMS AND CONDITIONS, YOU SHOULD CLICK ON THE 'REJECT' BUTTON, DELETE THE MATERIALS FROM YOUR COMPUTER AND PROMPTLY (AND IN ANY EVENT, WITHIN 14 DAYS OF RECEIPT) RETURN TO SOURCEGUARDIAN OR A LICENSED RESELLER (A) THE HARD MEDIA; (B) ANY OTHER ITEMS PROVIDED THAT ARE PART OF THIS PRODUCT; AND (C) YOUR DATED PROOF OF PURCHASE. ANY MONEY YOU PAID TO SOURCEGUARDIAN OR AN SOURCEGUARDIAN RESELLER FOR THE MATERIALS WILL BE REFUNDED LESS ANY CREDIT CARD TRANSACTION FEE INCURRED BY SOURCEGUARDIAN.

E OTHER AGREEMENTS

IF YOUR USE OF THESE PROGRAMS AND DATA IS PURSUANT TO AN EXECUTED LICENCE AGREEMENT, SUCH AGREEMENT SHALL APPLY INSTEAD OF THE FOLLOWING TERMS AND CONDITIONS.

LICENCE AGREEMENT AND LIMITED WARRANTY

1. Ownership of materials and copies

The Materials and related documentation are copyrighted works of authorship, and are also protected under applicable database laws. SourceGuardian retains ownership of the Materials and all subsequent copies of the Materials, regardless of the form in which the copies may exist. This licence is not a sale of the original Materials or any copies.

2. Licence

2.1. Evaluation Licence

If you have received an evaluation version of the Materials, SourceGuardian hereby grants to you, strictly for your own internal business purposes (and subject to the other terms and conditions of this Licence Agreement), a limited, non-exclusive licence for a single user to:

2.1.1. install the Materials for use on a single computer owned, leased and/or controlled by you for an evaluation period of 14 days;

2.1.2. make a single copy of the Materials for back-up, archival or other security purposes.

2.2. Full Licence

Provided that you have paid the applicable licence fee (and subject to the other terms and conditions of this Licence Agreement), SourceGuardian hereby grants to you, strictly for your own internal business purposes, a limited, non-exclusive licence for a single user to:

2.2.1. install the Materials for use on a single computer owned, leased and/or controlled by you;

2.2.2. make a single copy of the Materials for back-up, archival or other security purposes.

2.2.3. use SourceGuardian for development, testing, training and demonstration purposes and for the purpose of providing services to end users.

2.3. Subject to the remaining provisions of this Licence SourceGuardian grants to the Licensee a world-wide, royalty free, non-exclusive, licence to permit the Licensee to do the following things in relation to the Loader (being defined as the software program made available on the SourceGuardian website at www.sourceguardian.com in object code form that facilitates the conversion of scripts encoded with SourceGuardian to readable form). The following permissions shall be deemed to apply to and cover any use of the Loaders prior to the effective date of this Licence Agreement.

2.3.1. Distribute free of charge and make copies of the Loader for non-revenue generating activities including but not limited to evaluation, development, demonstration, training purposes, test, verification as well as for end user support. All such copies shall be subject to the provisions of this Licence Agreement;

2.3.2. Merge, incorporate, install and integrate the Loader with any third party or Licensee software;

2.3.3. Use, distribute and market the Loader to end users provided always that end users are either (i) directed to the SourceGuardian website and agree to the terms of SourceGuardian's free Loader Licence or (ii) are supplied with a copy of SourceGuardian's free Loader Licence when the encoded files are supplied.

3. Limited Support

SourceGuardian shall make available to you (at such times and to such extent as SourceGuardian may, in its sole discretion, deem reasonable) limited email support services for a period of 6 months from the date of your first installation of the Materials. Without prejudice to the foregoing provisions of this clause 3, such support services are, in any event, limited to your making a maximum of 20 requests for assistance during the support period.

4. Licence restrictions

You may not use, copy, modify or transfer the Materials (including any related documentation) or any copy, in whole or in part, including any print-out of all or part of any database, except as expressly provided for in this licence. If you transfer possession of any copy of the Materials to another party or use the Materials on a different computer from that on which the Materials were originally installed except as provided herein or without obtaining SourceGuardian's prior written consent, your licence is automatically terminated. You may not translate, reverse engineer, decompile, disassemble, modify or create derivative works based on the Materials, except as expressly permitted by the laws of England and Wales. You may not vary, delete or obscure any notices of proprietary rights or any product identification or restrictions on or in the Materials..

5. No transfer

The Materials are licensed only to you. You may not rent, lease, sub-license, sell, assign, pledge, transfer or otherwise dispose of the Materials, on a temporary or permanent basis, nor use the same for remote hosting, ASP services, to act as a bureau or for time-sharing use without the prior written consent of SourceGuardian.

6. Undertakings

You undertake to:

- 6.1. ensure that, prior to use of the Materials by your employees or agents, all such parties are notified of this licence and the terms of this Licence Agreement;
- 6.2. reproduce and include our copyright notice (or such other party's copyright notice as specified on the Materials) on all and any copies of the Materials, including any partial copies of the Materials;
- 6.3. hold all drawings, specifications, data (including object and source codes), software listings and all other information relating to the Materials confidential and not at any time, during this licence or after its expiry, disclose the same, whether directly or indirectly, to any third party without SourceGuardian's consent.

7. Limited warranty

- 7.1. Subject to the limitations and exclusions of liability below, SourceGuardian warrants that (a) the Hard Media on which the Materials are furnished will be free from material defects under normal use; and that (b) the copy of the program will materially conform to the documentation which accompanies the program. The Warranty Period is 90 days from the date of delivery to you.
- 7.2. SourceGuardian will also indemnify you for personal injury or death solely and directly caused by any defect in its products or the negligence of its employees.

7.3. SourceGuardian shall not be liable under the said warranty above if the Materials fail to operate in accordance with the said warranty as a result of any modification, variation or addition to the Materials not performed by the SourceGuardian or caused by any abuse, corruption or incorrect use or installation of the Materials, including use of the Materials with equipment or other software which is incompatible.

8. No other warranties

8.1. The foregoing warranty is made in lieu of any other warranties, representations or guarantees of any kind, either expressed or implied, including, but not limited to, any implied warranties of quality, merchantability, fitness for a particular purpose or ability to achieve a particular result. You assume the entire risk as to the quality and performance of the Materials. Should the Materials prove defective, you (and not the SourceGuardian nor any licensed reseller) assume the entire cost of all necessary servicing, repair or correction.

8.2. SourceGuardian does not warrant that the Materials will meet your requirements or that its operation will be uninterrupted or error free.

9. Limitation of liability

SourceGuardian's entire liability and your exclusive remedy shall be:

9.1. the replacement of any Hard Media not meeting SourceGuardian's 'Limited Warranty' and which is returned to SourceGuardian together with dated proof of purchase; or

9.2. if, during the Warranty Period, SourceGuardian is unable to deliver replacement Hard Media which is free of material defects, you may terminate this Licence Agreement by returning the Materials to SourceGuardian and any money you paid to SourceGuardian for the Materials will be refunded less any credit card transaction fee incurred by SourceGuardian.

10. Exclusion of liability

10.1. Except in respect of personal injury or death caused directly by the negligence of SourceGuardian, in no event will SourceGuardian be liable to you or any third party for any damages, including any lost profits, lost savings, loss of data or any indirect, special, incidental or consequential damages arising out of the use of or inability to use such Materials, even if SourceGuardian has been advised of the possibility of such damages. Nothing in this Licence Agreement limits liability for fraudulent misrepresentation.

10.2. Without prejudice to any other provisions of this Licence Agreement you hereby expressly acknowledge that encryption software is not infallible and that third parties may develop and employ methods to circumvent the Materials and you agree that SourceGuardian shall have no liability to you or any third party in such circumstances.

11. Your statutory rights

This licence gives you specific legal rights and you may also have other rights that vary from country to country. Some jurisdictions do not allow the exclusion of implied warranties, or certain kinds of limitations or exclusions of liability, so the above limitations and exclusions may not apply to you. Other jurisdictions allow limitations and exclusions subject to certain conditions. In such a case the above limitations and exclusions shall apply to the fullest extent permitted by the laws of such applicable jurisdictions. If any part of the above limitations or exclusions is held to be void or unenforceable, such part shall be deemed to be deleted from this Licence Agreement and the remainder of the limitation or exclusion shall continue in full force and effect. Any rights that you may have as a consumer (ie a purchaser for private as opposed to business, academic or government use) are not affected.

12. Term

The licence is effective until terminated. You may terminate it at any time by destroying the Materials together with all copies in any form. It will also terminate upon conditions set out elsewhere in this Licence Agreement or if you fail to comply with any term or condition of this Licence Agreement or if you voluntarily return the Materials to us. You agree upon such termination to destroy the Materials together with all copies in any form.

13. Export

You will comply with all applicable laws, rules, and regulations governing export of goods and information, including the laws of the countries in which the Materials were created. In particular, you will not export or re-export, directly or indirectly, separately or as a part of a system, the Materials or other information relating thereto to any country for which an export licence or other approval is required, without first obtaining such licence or other approval.

14. General

- 14.1. You agree that SourceGuardian shall have the right, after supplying undertakings as to confidentiality, to audit any computer system on which the Materials are installed in order to verify compliance with this licence Agreement.
- 14.2. This Licence Agreement constitutes the complete and exclusive statement of the Agreement between SourceGuardian and you with respect to the subject matter of this Licence Agreement and
supersedes all proposals, representations, understandings and prior agreements, whether oral or written, and all other communications between us relating to that subject matter.
- 14.3. Any clause in this Licence Agreement that is found to be invalid or unenforceable shall be deemed deleted and the remainder of this Licence Agreement shall not be affected by that deletion.
- 14.4. Failure or neglect by either party to exercise any of its rights or remedies under this Licence Agreement will not be construed as a waiver of that party's rights nor in any way affect the validity of the whole or part of this Licence Agreement nor prejudice that party's right to take subsequent action.
- 14.5. This Licence Agreement is personal to you and you may not assign, transfer, sub-contract or otherwise part with this Licence Agreement or any right or obligation under it without the SourceGuardian's prior written consent.

- 14.6. This Licence Agreement and any claim or matter arising under or in connection with this Licence Agreement and the legal relationships established by this Licence Agreement shall be governed by and construed in all respects in accordance with the law of England and Wales, and the parties agree to submit to the non-exclusive jurisdiction of the English courts.

Should you have any questions concerning this Licence Agreement you may contact SourceGuardian Limited at A6 Kinigfisher House, Kingsway, Team Valley Trading Estate, Gateshead, Tyne & Wear. NE11 0JQ United Kingdom. Tel: 0845 155 2455. Email: Support@SourceGuardian.com.

8.2 SourceGuardian Loaders License

This Licence applies to the use of the Loaders for SourceGuardian by End Users and is granted by SourceGuardian Ltd (registered number 05663267) which is referred to in this Licence as "SourceGuardian". The licence terms for the use of SourceGuardian software are set out at <http://www.sourceguardian.com/terms.html>. If you are a licensee of SourceGuardian your use of the Loaders is governed by that licence.

If you are an End User of SourceGuardian then, by downloading the Loader, you are accepting the terms of this Licence on behalf of yourself and any company, unincorporated association or partnership for which you work. This Licence comes into effect on the date that you download the Loader. If, having read the Licence, you do not agree to be bound to any of its terms you should not download the Loader and any enquiries on the content of this Licence should be directed to SourceGuardian Ltd at A6 Kinigfisher House, Kingsway, Team Valley Trading Estate, Gateshead, Tyne & Wear. NE11 0JQ United Kingdom. Tel: 0845 155 2455. Email: Support@SourceGuardian.com.

1. Definitions

Defect

Any material error that prevents the Loader from uploading or downloading scripts to and from SourceGuardian (unless used in conjunction with third party software that is not on the list of maintained software on SourceGuardian's website)

End User(s)

Users of the Loader in commercial operation

Loader

The software program that facilitates the conversion of scripts encoded with SourceGuardian to readable form

Maintainence

Issuing updates to SourceGuardian to ensure continuing compatibility with third party software programs with which SourceGuardian is designed to inter-operate.

SourceGuardian

The software encryption product of that name used to encrypt scripts from human- readable form into a form capable of being read only with the Loader, available at www.SourceGuardian.com

Support

Assisting the End User with enquiries relating to Defects

2. License for End Users

Subject to the remaining provisions of this Licence SourceGuardian grants to the End User a world-wide, royalty free, non-exclusive, licence to permit the End User to use the Loader in its commercial operations for the purposes of:

2.1 Reading scripts encrypted with SourceGuardian;

2.2 Bundling the End User's own software applications together with the Loader;

2.3 Linking the End User's software applications to the Loaders on SourceGuardian's website (to ensure that the applications remain current with the latest updated version of the Loader)

This License shall be deemed to apply to and cover any use of the Loaders prior to the effective date of this Licence.

3. Restrictions and Exceptions

3.1 The rights granted to the End User under this Licence do not operate to assign or transfer the ownership of any intellectual property rights in the Loader to the End User.

3.2 The Loader is intended for commercial use only and not for use by consumers. By accepting this Licence the End User confirms that he or she is acting in the course of business. The End User will not remove or obscure any copyright or ownership notices or warning legends from the Loader nor will the End User attempt to reverse engineer, decompile or otherwise interfere with the Loader except to the extent expressly permitted by law or under this Licence. SourceGuardian may terminate this Licence immediately if it discovers that the End User is in breach of its obligations in this Licence and in such a case the End User will immediately delete the Loader from its computer systems and will on request by SourceGuardian provide and execute such written assurances as SourceGuardian may require confirming such deletion.

3.3 The Loader is licensed free of charge and accordingly is provided on an "as is" basis. The End User agrees and acknowledges that SourceGuardian has no liability to the End User, whether in contract, tort (including negligence) or otherwise arising from any Defects in the Loader and all warranties implied by the laws of any jurisdiction in which the End User uses the Loader are expressly excluded to the fullest extent permitted by the laws of such jurisdiction.

3.4 In no event will SourceGuardian be liable to the End User for any consequential loss or any financial loss.

3.5 SourceGuardian's only obligation to the End User is to use reasonable commercial efforts to (i) correct Defects within a reasonable time (ii) ensure the the Loader is not corrupted and is free of any computer virus, trojans, worms or logic bombs and (iii) to issue Maintenance updates from time to time. SourceGuardian may terminate or assign its obligations under this paragraph 3.5 by publishing a notice to this effect on the SourceGuardian website at www.SourceGuardian.com. In such circumstances the provisions of paragraph 3.2 will also terminate. Nothing in this Licence applies so as to exclude or limit any liability that SourceGuardian may have to the End User based on fraudulent misrepresentation or personal injury resulting from SourceGuardian's negligence.

4. General

4.1 This Agreement contains the entire agreement between the parties on the subject matter of this Agreement and supersedes all representations, undertakings and agreements previously made between the parties with respect to the subject matter of this Agreement.

4.2 If any provision (or part of a provision) of this Licence is found by any court or administrative body of

competent jurisdiction to be invalid, unenforceable or illegal, the other provisions will remain in force. If any invalid, unenforceable or illegal provision would be valid, enforceable or legal if some part of it were deleted, the provision will apply with whatever modification is necessary to give effect to the commercial intention of the parties.

4.3 This Licence constitutes the whole agreement between the parties and supersedes any previous arrangement, understanding or agreement between them relating to its subject matter.

4.4 Each party acknowledges and agrees that in entering into this Licence it does not rely on any undertaking, promise, assurance, statement, representation, warranty or understanding (whether in writing or not) of any person (whether party to this Licence or not) relating to the subject matter of this Licence, other than as expressly set out in this Licence.

4.5 This Licence and any disputes or claims arising out of or in connection with it or its subject matter or formation (including non-contractual disputes or claims) are governed by, and should be construed in accordance with, the law of England and Wales.

4.6 The parties irrevocably agree that the courts of England have exclusive jurisdiction to settle any dispute or claim that arises out of or in connection with the Contract or its subject matter or formation (including non-contractual disputes or claims).

8.3 Other Licenses for SourceGuardian for OSX

This section includes licenses for other products used for creating SourceGuardian™ for macOS.

8.3.1 RegexKit Framework

Copyright © 2007-2008, John Engelhart

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- * Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- * Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- * Neither the name of the Zang Industries nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

8.3.2 Sparkle Framework

Copyright © 2006 Andy Matuschak

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

SourceGuardian 13 for macOS, Windows, Linux

Part



IX

9 Changelog

9.1 Version 13 / February 2022

Version 13 introduces encoding for PHP8.1 and updated GUI. We were glad to receive comments and suggestions and want to thank you very much for sharing your ideas! We are looking forward to hearing about other suggestions for improving SourceGuardian. Here is a list of our version 13 changes.

- Full support of PHP8.1 encoding including all the latest language features. Note, files encoded with older versions of SourceGuardian need to be re-encoded with SourceGuardian 13 in order to run under PHP8.1
- PHP8.1 language features including enums, readonly properties, first-class callable syntax, intersection types and more.
- New loaders for PHP8.1 Please check [our blog](#) and the [loaders page](#) for new versions.
- A fully rebuild GUI. You will not notice many visual changes but the updated version supports hi DPI monitors and has some other minor refinements.
- Communication with sourceguardian.com website uses SSL connection now.
- We removed PHP 4.3 to 5.2 versions support. They are too old to use and support.
- We removed support of 32-bit (x86) encoders and loaders for Linux and FreeBSD. Embedded platforms support did not change.

We are happy to work with our customers if they need bespoke loaders for other custom operating system. Please contact support@sourceguardian.com if you are interested.

We do not include loaders for all supported operating systems to the GUI package. Please visit out [loaders page](#) if you can't find the loader for your OS in the [Copy Loaders](#) dialog.

9.2 Version 12 / February 2021

Version 12 introduces encoding for PHP8 - a great new version of PHP for years. We were glad to receive comments and suggestions and want to thank you very much for sharing your ideas! We are looking forward to hearing about other suggestions for improving SourceGuardian. Here is a list of our version 12 changes.

- Full support of PHP8 encoding including all the latest language features. Note, files encoded with older versions of SourceGuardian need to be re-encoded with SourceGuardian 12 in order to run under PHP8.

- New fully supported PHP8 language features include: named parameters, attributes, union types, mixed pseudo type, null-safe operator, match expressions, using throw in expressions, constructor properties and others.
- Due to native support for attributes (annotations) in PHP8, PHP frameworks which uses annotations like Symfony must work with encoded files out of the box (after the frameworks get updated accordingly by their authors).
- New loaders for PHP8. Please check [our blog](#) and the [loaders page](#) for new versions

We are happy to work with our customers if they need bespoke loaders for other custom operating system. Please contact support@sourceguardian.com if you are interested.

We do not include loaders for all supported operating systems to the GUI package. Please visit out [loaders page](#) if you can't find the loader for your OS in the [Copy Loaders](#) dialog.

New Features

- Multiple external license file names may be specified when locking to a license. Separate license file names, paths or URLs by a comma in the --external or an appropriate GUI option. Please note, the license files will be checked in the order they are specified. You may mix local file names, paths or URLs. If neither of the license files is found and running of the protected file fails with a "a license file is required" error, the error message will include only the first license file name. This is normal and by design.
- `sg_get_const()` if called without parameters returns an array of all the constants.
- locking to a machine id is refined for Linux. Please note, Linux lacks of a strict machine id algorithm. If you are getting unstable machine id result when running protected scripts on Linux, consider locking to MAC address(es) instead of machine id as an alternative approach.

Update for PHP 7.x code

- We have fixed some critical issues in PHP 7.x family of encoders and loaders. This means files encoded with version 12 of SourceGuardian for PHP 7.x will not run with older loaders. If you use version 12 of SourceGuardian, please make sure you install and/or deploy the recent loaders.

9.3 Version 11.4 / February 2020

Version 11.4 introduces encoding for PHP 7.4. We were glad to receive comments and suggestions and want to thank you very much for sharing your ideas! We are looking forward to hearing about other suggestions for improving SourceGuardian. Here is a list of all the version 11.4 changes.

- Full support of PHP 7.4 encoding including all the latest language options. Note, files encoded with older versions of SourceGuardian need to be re-encoded with SourceGuardian 11.4 in order to run under PHP 7.4
- New loaders for PHP 7.4. Please check [our blog](#) and the [loaders page](#) for new versions

We are happy to work with our customers if they need bespoke loaders for other custom operating system. Please contact support@sourceguardian.com if you are interested.

We do not include loaders for all supported operating systems to the GUI package. Please visit out [loaders page](#) if you can't find the loader for your OS in the [Copy Loaders](#) dialog.

New Features

- New --ignore-symlinks option for the CLI version of the encoder to skip symlinks while recursively scanning the folders.

GUI updates

- Some visual changes for Lock and Preferences screens.

9.4 Version 11.3 / April 2019

Version 11.3 introduces encoding for PHP 7.3 and many new features. We were glad to receive comments and suggestions and want to thank you very much for sharing your ideas! We are looking forward to hearing about other suggestions for improving SourceGuardian. Here is a list of all the version 11.3 changes.

- Full support of PHP 7.3 encoding including all the latest language options.
- We fixed some issues with unexpected segfaults on PHP7+ and some other problems. If you experienced issues like that with your code, please re-encode with the latest version 11.3 of SourceGuardian and install the recent loader.

Note, files encoded with SourceGuardian 11.0, 10.x or older need to be re-encoded with SourceGuardian 11.3 in order to run them under PHP 7.3

- New loaders for PHP 7.3, updated loaders for older versions of PHP

We updated loaders and created new ones for PHP 7.3. Loaders for the following operating systems are available:

Windows 32-bit (VC6, VC9; VC11 PHP 5.5, 5.6; VC14 PHP 7.0, 7.1; VC15 PHP 7.2)
Windows 64-bit (VC11 PHP 5.5, 5.6; VC14 PHP 7.0, 7.1, VC15 PHP 7.2)
MacOSX (universal binaries, include i386, x86_64)
Linux (i386, x86_64)

We update the following loaders on request. Please check [our blog](#) and the [loaders page](#) for new versions.

FreeBSD (i386, x86_64)
Linux ARM (armel)
Linux ARM (armhf) Raspberry Pi (including Pi version 3) and other boards

We are happy to work with our customers if they need bespoke loaders for other custom operating system. Please contact support@sourceguardian.com if you are interested.

We do not include loaders for all supported operating systems to the GUI package. Please visit our [loaders page](#) if you can't find the loader for your OS in the [Copy Loaders](#) dialog.

New Features

- Experimental licensing for Docker. If you are installing SourceGuardian to your Docker, licensing must work correctly now letting you install and use SourceGuardian on this Docker machine. So, if you are using SourceGuardian from a Docker container during the deployment process of your files, this must work now since you installed and registered your copy of SourceGuardian as usual. Note, every separate installation of SourceGuardian still requires an additional license as one license lets you install and use SourceGuardian only on one machine.

Installing to Docker needs a special approach and mapping of /var/run/docker.sock Please refer to a new section [Installing to Docker](#) in the user manual.

- We are introducing a new locking option which is available in the full version of the encoder - locking to a machine ID. Use the new loader method `sg_get_machine_id()` to obtain a machine ID on your client machine and then specify the machine ID on the Lock screen in GUI or use the new command line option `--machine-id` of the encoder or `licgen` tool. Encoded PHP scripts locked to a machine ID will only run on that machine (or machines if you specify multiple machine IDs). Please refer to the [Locking options](#) section for further information.

Note: as `sg_get_machine_id()` is a binary method of the loader, an appropriate loader must be installed to the client machine in order for this method to be available from your code. We recommend that you create a mini project, encode it and run for obtaining the machine ID from the client.

- We are also introducing a new locking option for CLI PHP scripts. It was always a problem to lock such the scripts as neither IP nor domain locking may be used for them. Locking to MAC address was only a solution in that case, but it's not always convenient to lock to MAC addresses. Now you may use a special verification URL to validate the CLI script and make sure it works on the same machine as your web based part of the project.

For this to work you need to do two things:

1) Create a special PHP script which is accessible via HTTP protocol, it will be used to validate the machine. The only directive you need to put into it is:

```
<?php echo sg_get_verification_id(); ?>
```

Note: if you use any frameworks, you may need to use another mechanism for sending a string to the output instead of `echo`

2) When encoding your CLI script use the new `--remote-verification-url` option to specify the full URL to

your web verification script and use the same project ID as for the web part of your code and particularly the verification script.

Note 1: as the `--remote-verification-url` is mostly used only for locking of the encoded CLI scripts, consider separate encoding of the web part of your project and CLI scripts. You may create two GUI projects for that or use the command line encoder. Use the same project IDs/Keys for both!

Note 2: You may use standard locking to IP or domain for your web verification script (as usual for this to work your web server must send the information about current IP and domain to PHP via environment). Anyway, you "lock" to the domain if use the domain name in the verification URL or lock to IP if you use IP in the URL - choose the way which suits your project and the deployment process.

However, if your CLI PHP script works on its own and is not a part your your web based project, then you still may use the new [machine ID locking](#) option for it as well as good old locking to MAC addresses.

Please refer to the [Locking options](#) section for further information about using of the remote verification URL option.

- Option to add custom auto-globals. For CLI use: `--auto-global MYAUTO` If you use GUI, add this CLI option to 'addiitonal command line options' in [Advanced options](#).
- Options to ignore IP/domain check for the scripts running with CLI PHP. It means if you have encoded the entire project with locking to a IP/domain and selected the option to ignore the IP/domain check for CLI, your protected files will require to be run under the specified IP/domain(s) for web but the same files will be run OK with CLI PHP. This simplifies running PHP CLI scripts like cron jobs but still have them encoded in the same project along with web scripts. However, you may consider the risk of using this option as it does what it does - lets run your IP/domain locked encoded files with PHP CLI bypassing the domain check. For CLI use: `--ip-ignore-cli`, `--domain-ignore-cli` or the appropriate tickboxes on the [Lock screen](#) in GUI. Regardless these are the new options, please consider locking CLI scripts to the [verification URL](#) instead.
- PHP doc comments are now removed by default. Use `--keep-doc-comments` option to keep them e.g. if the framework you use require doc comments. We added an appropriate tickbox to the [Advanced options](#) window in GUI.
- Error text for `sourceguardian.restrict_unencoded=1` updated to include the file name

Error code [08] text changed to:
 PHP Fatal error: SourceGuardian Loader - This script ... or other unencoded scripts cannot run in this environment. Please contact the author of the script regarding this problem. [08]

Error code [15] text changed to:
 PHP Fatal error: SourceGuardian Loader - This protected script ... can run only in conjunction with other encoded files of the same project. Please contact the author of the script regarding this problem. Error code [15]

GUI updates

- File associations work now on Windows and macOS. It means clicking a .sg project file in Explorer/ Finder will launch SourceGuardian and opens this project in GUI. Note, for this to work, SourceGuardian must be installed using the installer on Windows, on macOS, you need to launch

SourceGuardian at least once for file associations to start working.

- Hex registration code is now displayed in Help/Registration information which is useful if you are using multiple installations of SourceGuardian and need to manage or reset a particular license in the online user profile, now you may easily find it there.
- You may access your SourceGuardian online user profile directly from the application, click Help/SourceGuardian User Profile.

CLI updates

- --days option was replaced with --expire 00d 00m 00h 00s
- New command line options described above: --ip-ignore-cli, --domain-ignore-cli, --machine-id, --machine-id-encrypt, --remote-verification-url, --keep-doc-comments, --auto-global

Bug Fixes

- using self:: from within nested function which in its turn is defined in a private member was not working

9.5 Version 11.2 / March 2018

Version 11.2 introduces encoding for PHP 7.2 and other new options for the command line encoder. We were glad to receive comments and suggestions and want to thank you very much for sharing your ideas! We are looking forward to hearing about other suggestions for improving SourceGuardian. Here is a list of all the version 11.2 changes.

- Full support of PHP 7.2 encoding including all the latest language options: converting numeric keys in object/array casts, counting of non-countable objects, object typehint, new optimized opcodes.
- We fixed some issues with unexpected segfaults on 7.1 and some other problems. If you experienced issues like that with your code, please re-encode with the latest version 11.2 of SourceGuardian and install the recent loader.

Note, files encoded with SourceGuardian 11.0, 10.x or older need to be re-encoded with SourceGuardian 11.2 in order to run them under PHP 7.2

- New loaders for PHP 7.2, updated loaders for older versions of PHP

We updated loaders and created new ones for PHP 7.2. Loaders for the following operating systems are available:

Windows 32-bit (VC6, VC9; VC11 PHP 5.5, 5.6; VC14 PHP 7.0, 7.1; VC15 PHP 7.2)
Windows 64-bit (VC11 PHP 5.5, 5.6; VC14 PHP 7.0, 7.1, VC15 PHP 7.2)
MacOSX (universal binaries, include i386, x86_64)

Linux (i386, x86_64)

We update the following loaders on request. Please check [our blog](#) and the [loaders page](#) for new versions.

FreeBSD (i386, x86_64)

Linux ARM (armel)

Linux ARM (armhf) Raspberry Pi (including Pi version 3) and other boards

We are happy to work with our customers if they need bespoke loaders for other custom operating system. Please contact support@sourceguardian.com if you are interested.

We do not include loaders for all supported operating systems to the GUI package. Please visit our [loaders page](#) if you can't find the loader for your OS in the [Copy Loaders](#) dialog.

Command line encoder updates

- A minor addition which may be a great help for the users who encode from the command line. We added "+" and "-" options for --phpversion. "+" means to encode for the specified version of PHP and for all the newer versions which are supported by the current version of SourceGuardian. "-" means to encode for all the supported versions of PHP except the specified one and all the lower versions, which is useful if you always need to encode by default for new versions but do not need support for old versions starting from some one. E.g.

--phpversion 5.6+	encodes for PHP 5.6 and all the newer versions (up to 7.2 for this version of SourceGuardian)
--phpversion 7.0+	encodes for PHP 7.0, 7.1, 7.2 (up to 7.2 for this version of SourceGuardian)
--phpversion 5.4-	encodes for PHP 5.5 and newer, i.e. excludes PHP 5.4 and older
--phpversion 4-	excludes PHP 4 completely, encodes for all PHP 5+, PHP 7+

As usual when encoding for multiple versions of PHP please make sure your code is compatible with ALL the selected versions of PHP, otherwise the encoder displays an error and that source PHP file may remain unencoded in the target folder.

9.6 Version 11.1 / April 2017

Version 11.1 introduces encoding for PHP 7.1, updated GUI and other new options. Although it looks like a minor update, in fact it's bigger than usual. We did serious work on refining GUI and adding some new and useful options to the command line version. As usual this update is partly based on comments and suggestions of our users. We were glad to receive comments and suggestions and want to thank you very much for sharing your ideas! We are looking forward to hearing about other suggestions for improving SourceGuardian. Here is a list of all the version 11.1 changes.

- Full support of PHP 7.1 encoding including all the latest language options: nullable types, void functions, class constants visibility, iterable, multi-catch and more.

Encoding for PHP 7.1 is fully supporting in version 11.1 of SourceGuardian. PHP 7.1 introduced new language features and updated bytecode format to support them. Files encoded with SourceGuardian 11.0, 10.x or older need to be re-encoded with SourceGuardian 11.1 in order to run protected files under

PHP 7.1

- New loaders for PHP 7.1, updated loaders for older versions of PHP

We updated loaders and created new ones for PHP 7.1. Loaders for the following operating systems are available:

Windows 32-bit (VC6, VC9; VC11 PHP 5.5, 5.6; VC14 PHP 7.x)
Windows 64-bit (VC9 PHP 5.3,5.4; VC11 PHP 5.5,5.6; VC14 PHP7.x)
MacOSX (universal binaries, include i386, x86_64)
Linux (i386, x86_64)

We update the following loaders on request. Please check [our blog](#) and the [loaders page](#) for new versions.

FreeBSD (i386, x86_64)
OpenBSD (i386, x86_64)
Linux ARM (armel)
Linux ARM (armhf) Raspberry Pi (including Pi version 3) and other boards

We are happy to work with our customers if they need bespoke loaders for other custom operating system. Please contact support@sourceguardian.com if you are interested.

We do not include loaders for all supported operating systems to the GUI package. Please visit out [loaders page](#) if you can't find the loader for your OS in the [Copy Loaders](#) dialog.

- We fixed some issues with string concatenation in PHP 5.6 and 7.0 and some other problems. If you experienced issues like that with your code, please re-encode with the latest version 11.1 of SourceGuardian and install the recent loader.

GUI updates

- We fully reworked the files and folders selection dialog which is used when you add files or folder to the project. This new dialog also uses predefined file filters which are based on your filter settings in File/Preferences.
- We fully reworked files and folders highlighting in the project tree. Folders - bold, virtual folders - green, files/folders changed or added since last encoding - blue.
- Newly added folders and files will be encoded at least once if the "encode only modified" option is selected in advanced settings.
- The option to encode only changed files (in Advanced Settings) always checks modification date for all the files in the project before encoding.
- The project is automatically checked for new and deleted files every time you open the project. This may be turned off/on in [Preferences](#).
- We added samples for code sections in [Advanced Options](#) - code for loader not installed, custom

PHP header and custom license text. Try the "Want sample" buttons after clicking "Edit" for the code you want to change.

- We added a new "Copy unencoded" filter section to [Preferences](#). These filters are checked before any further processing but after the exclusion filters. So, files that match any of "copy unencoded" file masks will be copied to the target folder as-is without encoding. E.g. you may now easily encode *.php files but keep *.tpl.php unencoded without manually selecting encoding mode in the project tree.
- If you are locking to a license file, now it's possible to select the folder where the license will be created when you click "Generate License" on the "Lock" screen. Also SourceGuardian will remind you to generate a license file after encoding, this option may be turned off/on in Preferences.
- We added tooltips to the main project window, advanced settings and some features on the lock screen to help our new users.
- You may automatically release the current SourceGuardian license directly from the application. Please find the "Release License" button added to Help/Registration Information. You will be asked for confirmation. Releasing the license lets you reinstall the encoder to another machine or to the same machine after upgrading hardware or OS. If you are going to upgrade the machine or OS, please firstly release the license and then you may transparently re-install your copy of SourceGuardian when the upgrade is complete.

You get 3 free license resets with the initial purchase. If you purchase an additional license or a new copy for another OS, each license also gets 3 resets. If you need to release the license after using all the 3 free resets, please [contact us in support](#).

- Note to Mac users. In this update we are launching the universal GUI also for Mac. It shares the code with Linux and Windows versions and it means it's easier to maintain and it sooner gets the new features and fixes. We will keep the native version of GUI for Mac for some time but it must be considered as discontinued. Both universal and native versions of GUI for Mac will be available for some time and some versions of SourceGuardian. We added support for PHP 7.1 for native GUI for Mac in this release.

Command line encoder updates

- We added automatic registration for command line tools. So, if you install the no-GUI package (Linux) or get an additional license for using command line tools on another machine, server etc e.g. to generate custom licenses there, you may use automatic registration on the first run. Run sourceguardian command line executable or licgen tool, read and agree with the terms, enter your SourceGuardian online user account email and password when asked in the terminal. If automatic registration can't be used for any reason, e.g. if there is no Internet access, register software as usual. Copy the hex code from the screen, paste it to the user profile, download the encode.lic license file and copy it to the binary folder where the executable is located.
- The command line tools now may be started with GUI license. It's not necessary to specify a path to the GUI encode.lic license file anymore using the -L option. The GUI license will be automatically found if you are starting the command line tools included to GUI installation. However, if you are installing the command line tools separately or to another machine, please register your copy as usual and install the encode.lic license file to the same binary folder where the tools are located or use automatic registration for new copies.
- The license may be released automatically from for the command line tools as well. Please read

above as we added the same option to the GUI version. In order to release the license from the command line, use `--license-release` option with sourceguardian command line tool or licgen and follow the instructions displayed on your terminal.

- We added a new `-c` (`--copy`) filter option in addition to `-f` (`--file`), `-t` (`--template`) and `-x` (`--exclude`). The new `-c` (`--copy`) option may be used to specify the files that will be copied as-is without encoding to the target folder. This option makes sense only if you specify the target folder with `-o` (`--output`) option. If `-c` is used without `-o`, then it works as `-x` and skips the specified files. The new option may take `*` and `?` wildcards or a `@filelist`.

E.g. you may now encode `*.php` files but keep `*.tpl.php` unencoded and copy the latter ones as-is:

```
/path/to/sourceguardian -o /path/to/target -f "*.php" -c "*.tpl.php" /path/to/source
```

- Optional directory trimming level may be specified with `-r{n}`. The default is 0 and means no trimming, this matches the mode used in previous versions of SourceGuardian. If `n` is specified, the encoder will remove `n` folder names from the beginning of target file paths when encoding or copying the files to the target folder. This is similar to `-p` option of patch utility on Unix. You may find some [samples](#) useful.
- We updated recursive directory search. Source paths containing wildcards in directory names now work, e.g. `/path/to/dir?/*.php`. This also works in `@filelist` and you may use it with `-f`, `-t`, `-c`, `-x`. If the `@filelist` is specified, recursion is automatically turned on, but it's still possible to change the directory trimming level with `-r{n}` if necessary.
- `{SG_EXPIRY_DATE}` tag was added and may be used in the custom text within the generated license files. Use this tag to embed your custom license expiry date into the readable part of the license. Changing of this text in the license file does not make sense and is NOT opening a back door, if you use the expiry date in your custom licenses. The expiry date is stored encrypted as well as other options within the license file. The readable text is only for information and you may put a name of your product, name of your customer and now the expiry date. For further details please see [Advanced Options](#) in GUI or `--text` option for the [command line licgen tool](#).

9.7 Version 11.0.6 / October 2016

Version 11.0.6 is a minor update release that includes some fixes and updates. We were glad to receive comments and suggestions and want to thank you very much for sharing your ideas! We are looking forward to hearing about other suggestions for improving SourceGuardian.

FIXES

- Fixed `__DIR__` and `__FILE__` constants that were statically compiled when used in initialization expressions in PHP5.6 (older versions of PHP do not allow to use constants in initialization expressions, PHP7 was already handled correctly). If you get issues with using `__DIR__` or `__FILE__` in your PHP 5.6 code, re-encode the files with version 11.0.6 of SourceGuardian and install the updated loaders.
- Fixed `PHP_OS` and some other constants were statically compiled for PHP7. If you get issues with using `PHP_OS` or other constants in your PHP7 code, re-encode with SourceGuardian 11.0.6.

UPDATES

- We updated how empty files are processed. Now empty files are copied to the target folder if it's specified. Empty files are skipped from processing if the target folder is not specified. In either case empty files are not counted as errors anymore. The encoding log will still indicate empty files with [11] code.
- We added the --verbose option to the command line encoder. Options are 0-quiet, 1-print only errors in the log, 2-print standard log. 2 is default. You may also add this option to "Additional Command Line Options" in "Advanced settings" if you want to change the encoding log when using GUI.
- The GUI license generator must correctly handle http:// network paths to the license. In that case the license is created in the target folder under the specified URL path, the http:// and domain name are filtered to build a correct local path to the license.
- Updated loaders for PHP 5.6 and 7. Please check the [loaders page](#) for new versions.

9.8 Version 11 / June 2016

Version 11 introduces encoding for PHP 7.0 and some new options. As usual this update is partly based on comments and suggestions of our users. We were glad to receive comments and suggestions and want to thank you very much for sharing your ideas! We are looking forward to hearing about other suggestions for improving SourceGuardian. Here is a list version 11 changes.

NEW FEATURES

- Full support of PHP 7.0 encoding including the latest language options: scalar type declarations, return type declarations, new operators, anonymous classes and more.

Encoding for PHP 7.0 is fully supporting in version 11 of SourceGuardian. PHP 7.0 introduced new language features and updated bytecode format to support them. Files encoded with SourceGuardian 10.x or older need to be re-encoded with SourceGuardian 11 in order to run protected files under PHP 7.0

- New loaders for PHP 7.0, updated loaders for older versions of PHP

We updated loaders and created new ones for PHP 7.0. Loaders for the following operating systems are available:

Windows 32-bit (VC6, VC9; VC11 PHP 5.5, 5.6; VC14 PHP 7.0)
 Windows 64-bit (VC9 PHP 5.3,5.4; VC11 PHP 5.5,5.6; VC14 PHP7.0)
 MacOSX (universal binaries, include i386, x86_64)
 Linux (i386, x86_64)

We update the following loaders on request. Please check [our blog](#) and the [loaders page](#) for new versions.

FreeBSD (i386, x86_64)
 OpenBSD (i386, x86_64)
 Linux ARM (armel)

Linux ARM (armhf) Raspberry Pi (including Pi version 3) and other boards

We are happy to work with our customers if they need bespoke loaders for other custom operating system. Please contact support@sourceguardian.com if you are interested.

We do not include loaders for all supported operating systems to the GUI package. Please visit our [loaders page](#) if you can't find the loader for your OS in the [Copy Loaders](#) dialog.

- We removed the @SourceGuardian tag from all the encoded files excepting files encoded with so-called "[conjunction](#)" option.
- Locking to a domain automatically enables www. subdomain and you do not need to specify the www. subdomain on the locking page.
- The command line tools now return expected exit codes. The encoder returns encoding status after processing a single file. When it is processing multiple files it returns zero in case of running the process and if there are no issues in using the command line options and then you need to check the encoding log for further details. Licgen returns exit code on invalid options or status of the license generation. Sginfo returns exit code on invalid options or status of the encoded file. Please find further details in the [Exit codes](#) section.
- A built-in file information tool or its command line equivalent now returns estimated memory usage for running the encoded file. This does not include any additional memory requirements of your code. Estimated memory usage also differs for versions of PHP as lengths of bytecode may vary.
- A new `sg_loader_version()` function returns version of the loader. As well as any other loader functions it's available only when the loader is loaded into PHP, i.e. you may use it from your protected files or if the loader is installed to `extension_dir` and `php.ini`.

9.9 Version 10 / June 2014

Version 10 introduces improved code protection methods as well as encoding for PHP 5.6 and some new options. As usual this update is partly based on comments and suggestions of our users. We were glad to receive comments and suggestions and want to thank you very much for sharing your ideas! We are looking forward to hearing about other suggestions for improving SourceGuardian. Here is a list of recent version 10 changes.

NEW FEATURES

- Improved code protection methods
- Full support of PHP 5.6 encoding including the latest language options: constant scalar expressions, variadic functions operator, updated `use` operator and more

Encoding for PHP 5.6 is fully supporting in version 10 of SourceGuardian. PHP 5.6 introduced new language features and updated bytecode format to support them. Files encoded with SourceGuardian 8, 9.x or older will need to be re-encoded with SourceGuardian 10 in order to run protected files under PHP 5.6

- New loaders for PHP 5.6, updated loaders for older versions of PHP

We updated loaders and created new ones for PHP 5.6. Loaders for the following operating systems are available:

Windows 32-bit (VC6, VC9; VC11 PHP 5.5, 5.6)
Windows 64-bit (VC9 PHP 5.3,5.4; VC11 PHP 5.5,5.6)
MacOSX (universal binaries, include i386, x86_64)
Linux (i386, x86_64)
FreeBSD (i386, x86_64)

We update the following loaders on request. Please check [our blog](#) and the [loaders page](#) for new versions.

OpenBSD (i386, x86_64)
IBM PowerLinux
HP-UX Itanium
Linux ARM (armel)
Linux ARM (armhf) including Raspberry Pi

We are happy to work with our customers if they need bespoke loaders for other custom operating system. Please contact support@sourceguardian.com if you are interested.

We do not include loaders for all supported operating systems to the GUI package. Please visit out [loaders page](#) if you can't find the loader for your OS in the [Copy Loaders](#) dialog.

- A closing PHP tag is not added anymore to the end of encoded files. It's not needed, Zend recommends to not use it. Omitting the closing tag 'automatically' protects against 'headers already sent' errors if any of the encoded files were accidentally opened/saved in the editor and some characters were added at the end. Omitting the closing tag does not affect execution of protected files.
- SG_LIC_PATH environment variable may be used to specify where the loader should search for a license file. See details for [GUI](#), for [command line](#)
- PHP short tags <? ?> are enabled by default. If you do not need them enabled for any reason, you may turn them off in [advanced settings in GUI](#) or by using the new [--no-short-tags option in the command line](#). The old --short-tags option has been removed.
- License generation fixed in GUI when a URL is specified as a license file name.
- Fixed how encoding only of changed files works in GUI.
- A new 'Refresh' button added to the GUI for updating the project tree. It is useful if any of the files were changed or added to the folders behind the encoder GUI. Also automatic refresh happens when you encode only files updated since the last encoding (see [Advanced settings](#)).
- A new --keep-file-date [command line option](#) added to keep the modification date for encoded files the same as the modification date of source files. The same option is available in GUI in [Advanced settings](#).

9.10 Version 9.5 / July 2013

Version 9.5 introduces encoding for PHP 5.5 and adds some new options. As usual this update is partly based on comments and suggestions of our users. We were glad to receive comments and suggestions and want to thank you very much for sharing your ideas! We are looking forward to hearing about other suggestions for improving SourceGuardian. Here is a list of recent version 9.5 changes.

NEW FEATURES

- Full support of PHP 5.5 encoding including the latest language options: generators, coroutines, 'finally' operator and more

Encoding for PHP 5.5 is fully supporting in version 9.5 of SourceGuardian. PHP 5.5 introduced new language features and updated bytecode format to support them. Files encoded with SourceGuardian 8, 9.0 or older will need to be re-encoded with SourceGuardian 9.5 in order to run protected files under PHP 5.5

- New loaders for PHP 5.5, updated loaders for older versions of PHP

We updated loaders and created new ones for PHP 5.5. Loaders for the following operating systems are available:

- Windows 32-bit
- Windows 64-bit (PHP 5.3,5.4,5.5)
- MacOSX (universal binaries, include i386, x86_64)
- Linux (i386, x86_64)
- FreeBSD (i386, x86_64)
- OpenBSD (i386, x86_64)
- IBM PowerLinux
- HP-UX Itanium
- Linux ARM (armel)
- Linux ARM (armhf) including Raspberry Pi

We are happy to work with our customers if they need bespoke loaders for other custom operating system. Please contact support@sourceguardian.com if you are interested.

We do not include loaders for all supported operating systems to the GUI package. Please visit our [loaders page](#) if you can't find the loader for your OS in the [Copy Loaders](#) dialog.

- Standard input and output support for the [command line encoder](#).
- New option to disable additional CRC check to enable eval()'ing protected code. See details for [GUI](#), for [command line](#)
- Added "Script will expire in (days)" option to the [Locking page](#).

- Changed how 'modification date' works in GUI
- The 'Copy unencoded' mode is now assigned to empty files when adding them to the project. This is to eliminate 'empty file - skipped' error messages from the encoder. It replicates what GUI does with other files that are not detected by their file extensions. If you don't need to copy empty files to the target folder, you may change the encoding mode to 'Skip' for them.
- Reverting the project was affecting modification dates - fixed
- Updated built-in support and automatic update in GUI.

9.11 Version 9.0 / July 2012

Version 9.0 introduces encoding for PHP 5.4, fixes problems and adds some new options. The update is partly based on comments and suggestions of our users. We were glad to receive comments and suggestions and want to thank you very much for sharing your ideas. We are looking forward to hearing about other suggestions for improving SourceGuardian and we are open to new ideas. Here is a list of recent version 9.0 changes.

NEW FEATURES

- Full support of encoding for PHP 5.4

Encoding for PHP 5.4 is fully supporting in version 9 of SourceGuardian. PHP 5.4 introduced new language features and updated bytecode format to support them. Files encoded with version 8 or older of SourceGuardian will need to be re-encoded with SourceGuardian 9 in order to run protected files in PHP 5.4

- New loaders for PHP 5.4, updated loaders for older versions of PHP

We updated loaders and created new ones for PHP 5.4. Loaders for the following operating systems are available:

- Windows 32-bit
- Windows 64-bit (PHP 5.3,5.4)
- MacOSX (universal binaries, include i386, ppc, x86_64, ppc64)
- Linux (i386, x86_64)
- FreeBSD (i386, x86_64)
- OpenBSD 5.0,5.1 (i386, x86_64)

We may try to compile bespoke loader for other custom operating system. Please contact support@sourceguardian.com if you are interested.

- New option to stop encoding on E_DEPRECATED PHP 5.3+ compiler errors. See details for [GUI](#), for [command line](#)
- New option to encode only changed files detected by file modification date. See details for [GUI](#), for

[command line](#)

- Allow absolute file system path to a custom license file or an URL. [See details](#)
- Improved compatibility for scripts encoded for PHP 5.2.x
- Automatic filtering UTF-8 BOM from all source files. This allows encoding files created in editors that save BOM.
- Protected scripts starter code has been reworked and improved. The default starter code now recognizes "thread safety" option providing a correct loader name for systems running thread safe PHP. It also displays instructions to a user about what loader is needed, where to download it and how to install it. The old version was displaying the loader name and the error message. It is still possible to exclude the default starter code from protected scripts (using -n command line encoder option or an appropriate option in GUI). Also it is still possible to replace the standard error message about a missing loader (using -j command line option or an appropriate option in GUI) with a custom one which may be either html/text or php code. If it's PHP code a new \$__ixedurl variable may be used, it contains the required loader download link.

For information: The starter code is a non-encoded portion of php code that is added by default to encoded PHP scripts. The task of the starter code is to detect if SourceGuardian loader is present and if it is not, then try to load it automatically if PHP configuration allows to do it. If automatic loading is not possible, the starter code will display a message telling a user what loader is needed, where to download it and how to install it in order to run SourceGuardian protected files.

- Now it is possible to send licgen's output to console instead of a file, use -- (double dash) instead of the output file name. [See details](#)
- Custom constants substitution in the custom header code. All user {constants} will be replaced in the prepend code. Also some standard SG constants may be used:

{SG_DATE} - current date i.e. date of encoding

{SG_LICENSEE} - SG license owner from the SG license file

It works in the same way also for licgen and do replacements for custom text if it is used.

BUGFIXES

- Fixed displaying strict errors (--strict-errors) that did not work if displaying deprecated errors (--deprec-errors) was not specified.
- On multiple compiling error the encoder was displaying the last one, while PHP displays the first one. Fixed.
- Updated protected scripts header. A protected script will not stop execution in case of 'extension_dir does not exists' error, the script will continue and fails in a standard way or an assigned error handler function will be called. This allows to catch errors like that in the custom error handler.
- Fixed encoding of PHP predefined constants like DIRECTORY_SEPARATOR in PHP 5.3+ (re-encoding is required). Protected scripts always do dynamic lookup as values of standard constants may differ on systems depending on OS, CPU etc.
- Fixed running domain locked scripts on web servers returning a port number in HTTP_HOST/ SERVER_NAME (e.g. mydomain.com:88)

- Fixed APC and other PHP bytecode cache compatibility issue in loaders.
- Now it is possible to catch SourceGuardian errors like "SourceGuardian Loader - script header is broken [10]" etc with a standard PHP error handling mechanism.
- Fixed --short-tags option which was always on in the previous versions of the encoder.

MacOSX GUI

- Two Project Window modes: "standard" and "file manager". This may be changed in Preferences, please read further details in Preferences / Interface & Updates.
- More informative log. Added total files counter, copied files counter and list, additional warning if the encoder has not completed encoding successfully.
- Exclusions in Preferences was changed to file masks instead of file extensions.
- Project window. "Don't encode" type renamed to "Copy unencoded", added new "Skip" type which are more straightforward.
- Support of the new options of version 9 encoder (PHP 5.4 encoding, new loaders, new advanced options etc)

NEW GUI FOR WINDOWS AND LINUX

- We are proud to present a fully reworked new cross platform GUI for Windows and Linux. It includes all the features of GUI for OSX and even more. It has built-in support system and more.

SUPPORTED PHP VERSIONS

- Encoding for PHP 4.3.x to PHP 5.4.x are fully supported

SUPPORTED OS

- Encoder is available for Windows, MacOSX, Linux (i386 and x86_64 versions).
- GUI and command line encoders and tools are included.

9.12 Version 8.2 / April 2010

Version 8.2 is partly based on comments and suggestions of our users. We were glad to receive them and want to thank you very much for sharing your ideas. We are looking forward to hearing about other suggestions for improving SourceGuardian and we are open to new ideas. Here is a list of recent version 8.2 changes.

- **Added a new predefined loader constant 'loader_version'**

`sg_get_const('loader_version')` returns a loader version number when called from the protected code.

- **Added a new license generator option to add custom text into the license file.**

We have added an option to include custom text into license files generated by SourceGuardian license

generator. The included text is protected with a checksum against modification. The option may be used to include user information, license description etc into license files.

- **API function `sg_get_const()` issue fixed.**

In previous versions it was possible to get an encoded constant value from the unencoded code included from the protected one. It has been fixed.

- **The last catch block issue fixed.**

The last catch block did not terminate the execution of the code in previous versions. If you are experiencing this problem you need to re-encode your code with version 8.2 of the encoder and install the updated loader.

- **Mac GUI improvements.**

We have added an exclusion list to Preferences which lets exclude files or folders that should not be included into the project when you add files. You can easily exclude .svn folders for example.

Added an option to encode only selected folders or files - you need to press Command key while clicking the Encode button. This option is useful if you need to re-encode only some files without doing it for the entire project.

Added Custom license text in Advanced settings which reflects a new generator option mentioned above.

New Preferences screen.

Other minor fixes.

- **Documentation update.**

The loaders installation section has been revised, added important information for Windows users. Other minor changes have been done reflecting the improvements mentioned above.

9.13 Version 8.1 / January 2010

Version 8.1 is an update based mainly on comments and suggestions of our users. We were glad to receive them and want to thank you very much for sharing your ideas. We are looking forward to hearing about other suggestions for improving SourceGuardian and we are open to new ideas. Here is a list of recent version 8.1 changes.

- **New loader API functions for encryption and decryption of a string.**

We have added new loader API functions `sg_encode_string()` and `sg_decode_string()` which can be used for encryption and decryption of a string when called from the protected PHP code. See [SourceGuardian loader API](#) section for details.

- **New algorithm for searching for the loader**

We have changed the way protected scripts search for loaders when dynamic loading is possible. Now protected scripts use the following rules:

<u>Version</u>	<u>Logic</u>
PHP 5.2.5+	Loader is loaded? If no, try extension_dir, then custom error or standard message.
<PHP 5.2.5	Loader is loaded? If no, try to dl() from the current dir then parents, if not found - try extension_dir. If still not found, then custom error or standard message.

Note: dynamic loading is only possible from extension_dir since PHP 5.2.5 and we do not try any other directories for PHP 5.2.5+

- **Option to stop encoding at a critical error**

We have added a new option for the encoder to stop at a critical error. See [GUI Manual / Project / Advanced Options](#) and [Using the command line encoder / Advanced Options](#) sections for details.

- **Option to report E_STRICT compiler errors**

E_STRICT "Strict Standards" warnings were introduced in PHP 5. Now the encoder has an option to warn of such messages during encoding. See [GUI Manual / Project / Advanced Options](#) and [Using the command line encoder / Advanced Options](#) sections for details.

- **User files encoding API changes**

We have changed the way how script keys based on Project ID or Project Key (if an external license file is used) are used for encoding or decoding files with sg_load_file() and sg_encode_file() loader API functions. Now we keep track of which protected PHP script uses which key and the loader will use an appropriate key for encryption/decryption of user files and strings (see new sg_encode_string() and sg_decode_string() API functions). Older versions of loaders would use the key of the last loaded protected script. These changes allow to use the encoding functions correctly in a situation when one protected script is inherited from another protected script and the above encoding/decoding functions are called. See [Creating custom encoded files from protected scripts](#) section for details of using these functions.

- **Marking a file to be skipped during encoding**

It's possible to mark a file to be skipped by the encoder. Add the following string anywhere in the code, use comments. Skipped files will be copied as-is to the target folder if it's specified.

SourceGuardian:DO_NOT_ENCODE

E.g. /* SourceGuardian:DO_NOT_ENCODE */

Note: Comparison is case sensitive for Windows and Solaris if you will ever move to any of these platforms for encoding your source files. Do not change the case for better code compatibility.

- **IPv4 address mapped to IPv6 is correctly recognized**

The loaders now correctly understand IPv4 mapped to IPv6 ::ffff:XX.XX.XX.XX style IP addresses in `$_SERVER['SERVER_ADDR']`. This fixes issues with locking to IP when a webserver returns IPv6 style address.

Note: SourceGuardian does not support locking to IPv6 yet, only to IPv4 addresses.

- **Tested compatibility with xdebug extension**

We have tested compatibility of loaders and xdebug PHP debugging extension. There is NO any compatibility issue except for the scripts encoded with `--conj` (script will only work with other encoded files) option. As xdebug modifies the bytecode files encoded with this option will fail with a conjunction error (code [15]). It's NORMAL behaviour demonstrating one of the points of protection. Obviously the internals of the encoded script cannot be debugged so once the debugger reaches the `sg_load()` function the entire protected script will be executed in one step.

- **Tested compatibility with APC**

We have tested compatibility of loaders and APC PHP cache extension. APC v3.0.19 the latest stable was tested. All protected scripts get cached no problem.

- **Automatic update of the project's contents for macOS GUI**

The contents of a project (files and folders added to the project) will be automatically synchronized with the contents of real folders on a disk when the project is reopened in SourceGuardian. If there are any changes within the added folders then the inner list of files and folders will be recursively updated. If there are any changes of the folders or files which were directly added on the top level then the application will query whether to remove missed elements or leave them within the project.

Note: Newly added files will get an encoding type according to their file extensions and settings in preferences. Probably you will want to check the type setting for new files.

Index

- E -

encode.lic 44

Error messages during protected scripts run 90

- F -

First run 44

- L -

License 44

Loader errors 90

- R -

rubyencoder 44