ionCube PHP Encoder 6.5

User Guide

ionCube Ltd. www.ioncube.com

CONTACTS AND LINKS

Contacting ionCube

We can be contacted as follows.

ionCube UK Headquarters

2A Exbury Road
Catford
London, SE6 4ND
United Kingdom

Telephone: +44-(0)208-690-6111

e-mail contacts
Enquiries: info@ioncube.com
Sales: sales@ioncube.com

IonCube Japan

5F 4-11-1 Hongo Bunkyo
Tokyo
113-0033
Japan

Telephone: +81-(0)3-5976-6931

e-mail contacts
Enquiries: info@ioncube.jp
Sales: sales@ioncube.jp

ionCube Poland

ionCube / NEO.pl
ul. Gen. Dabrowskiego 38-40
70-100 Szczecin
Poland

Telephone: +48-(0)91-4324211

e-mail contacts
Enquiries: info@ioncube.pl
Sales: sales@ioncube.pl

FAQ

Find answers to common questions in our FAQ at www.ioncube.com/faq.php

Support

For online support please visit support.ioncube.com or our online forum at forum.ioncube.com

Purchasing Products

To purchase ionCube products please visit www.ioncube.com/purchase.php

Errors, Omissions and Suggestions

In common with all ionCube products, the utmost care and attention has been put into the preparation and checking of this User Guide. In the unlikely event that you discover any errors or omissions, or should you have suggestions for improvements, please contact our publications department at publications@ioncube.com who will be pleased to hear from you.

CONTENTS

1 INTRODUCTION

The ionCube PHP Encoder is a powerful, high performance solution for encoding and licensing PHP scripts and encrypting files of any type.

Encoding and Encryption Total Solution

The Encoder protects PHP /HTML scripts with obfuscated bytecode protection and a custom execution engine. In addition, any other project files can be automatically *encrypted* if required, which is ideal for protecting files such as templates or XML documents. This is complemented by Loader API functions for reading and writing encrypted files. For most existing template engines, a small change is all that would be required to add the ability to read encrypted templates, and for Smarty users, we have a ready made patch to enhance Smarty at www.ioncube.com/resources/encrypted_smarty_patch.phps

Bytecode Compilation and Obfuscation

The Encoder achieves script protection by first compiling and then optimising PHP scripts to highly efficient binary data. This compilation process replaces source with virtual-machine instructions and applies several layers of encoding and transformations to produce the final platform independent files. Optional function name and local variable binary obfuscation adds extra protection, with further internal obfuscations applied at compile and runtime. This approach has the advantage that files are never restored to PHP source, compiled code is changed and hidden from the Open Source PHP engine, and gives excellent run-time performance due to parsing and compilation taking place at encoding time. Other Encoder features allow for the easy addition of plain text to the start of files, which is ideal for including custom copyright or license details, and is protected against tampering.

License File Creation

Create license files for your projects with the Pro and Cerberus Encoders to lock your projects to particular machines with optional time expiry. Applications can also have custom behaviour based on license file properties if required.

ionCube Loader

The ionCube Loader supports execution of encoded PHP files, encrypting or decrypting general files, validating licenses, and so on. This component can be easily installed into the php.ini file, or on many servers, automatically installed when required by the scripts themselves. This is ideal for shared servers where users often do not have access to the `php.ini` file. The Loader is also compatible with other popular extensions, and tools are provided to assist installation.

Windows GUI

For Windows users, a powerful GUI, described online and in the package, makes setting up projects a breeze. As well as the expected encoding features, integration with Explorer adds usability features such as dynamic icons to distinguish between encoded and unencoded files at a glance, and quick right click encoding with a minimal cut-down GUI. Upgrade to the Special Edition GUI for features such as automatic archive creation, FTP, and the unique *dynamic fields* feature that dramatically simplifies data entry for custom encoding and license creation through a dynamically created custom interface. Together with other features, the GUI helps to deliver the best productivity and flexibility possible.

1.1 Encoder Outline

The Encoder is driven by a command line interface allowing for easy automation and integration into project build, test and release environments and for calling from UNIX shell scripts or Windows batch files. Encoding files is fast, making it practical to perform "just in time" (JIT) encoding. This can be useful for producing trialware or customised versions of your software.

Key features include:

- Maximum protection and performance with bytecode encoding of PHP, combined PHP/HTML files and PHP shell scripts.
- File encryption feature plus Loader API support for reading/writing encrypted data, ideal for encrypting templates and XML documents.
- Choice of secure ASCII encoded file format for reliable cross platform transfers with FTP or binary file format.
- Optional binary obfuscation of function and local variable names
- Recognition of `<?php` and `<?` tags.
- Single file or recursive directory encoding
- File and attribute replication.
- Full control over the project items to be encoded, encrypted, copied or ignored.
- Custom text such as copyright details may be added to encoded files.
- Optional *include-attack* protection to block interaction with unauthorised files.
- Associating key/value *properties* with files.
- Projects feature to encode projects with pre-configured options.
- Custom Loader event messages or error handler callback.
- Automatic syntax checking and error reporting of encoded files.
- Syntax-check only operation.
- Exceptionally fast encoding performance.

Features of the Pro Encoder include Basic Encoder features plus:

- Generating files to expire after a time period or on a specific date.
- Generating files restricted by IP addresses and/or domain names.
- Generating files to work only with a valid license file.
- License generator program for creating license files with time expiry, machine restrictions and custom properties.

Features of the Cerberus Encoder include Pro Encoder features plus:

- Generating files restricted to Ethernet (MAC) addresses.
- Creating license files with MAC address restrictions.

1.2 User Guide Notation

1.2.1 Command Examples

Command or program related text in the user guide is printed in `monospace` type. Command examples are printed in the form:

```
ioncube_encoder source.php -o target.php
```

or to illustrate command usage and output, as

```
$ ioncube_encoder -V

ionCube Encoder Version 6.5
Copyright (c) 2002-2006 ionCube Ltd.
```

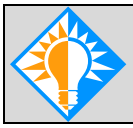
Shell input is shown in **`bold`**, and program output is `plain`. **`$`** is an example shell prompt for command entry but this may be different on your own system.

1.2.2 PHP 4 and 5 Encoders

In this document we shall refer to the Encoder command line program executable as `ioncube_encoder`. This program is the PHP 4 Encoder, and it should be used if encoding a PHP 4 project to ensure that your encoded scripts will run on PHP 4 and PHP 5 servers. The program `ioncube_encoder5` is the PHP 5 Encoder, and should be used when scripts contain language features only supported in PHP 5. For maximum compatibility, the PHP 5 Encoder supports the language syntax of PHP 5.0 and above.

1.2.3 Hints and Tips

Look out for hints and tips in the guide, for example:



You can use the bookmark feature in Adobe Acrobat to quickly navigate the user guide, or click on items in the contents section.

2 GETTING STARTED

This chapter introduces some of the most common features of the ionCube command line Encoder, with quick-start examples for typical scenarios. For Windows users, an additional document describes the Windows GUI. Chapter 3 will then describe all command line options in detail, and will be a useful reference guide to features even if using the Windows GUI. The License generation program is described in chapter 4, and chapter 5 describes the Loader API, including features for reading and writing encrypted files.

Remember that if reading this guide using Acrobat, you can use the Acrobat bookmarks feature to quickly jump to different sections, and in the document you can click on hyperlinks, entries in the contents section, and on any section references.

2.1 Licensing the Encoder

Each Encoder license is associated with the machine identity. The first step in licensing is to request a license for the machine where you wish to run the Encoder.

2.1.1 Generating a License Request

Run the Encoder with the `--license-request` option. This will attempt to create a license request for the Encoder. Your license will be partly tied to the MAC address of one of your network adapters, and so the Encoder will create a license request file for each network interface installed into your computer. It will also display the name of each interface found and the name of the file it created containing the license request. If the Encoder was unable to create license request files then the request data will be displayed instead. Your network interfaces do not need to be connected to any network but should always be present. Licensing to a wired adapter is recommended in preference to a wireless adapter.

What if my IP Address Changes?

This is not a problem. Although the license request files show an IP address, this is only help you identify the adapter that you want to license to. IP address is not part of the license data.

Example:

```
$ ioncube_encoder --license-request

Generating license request for computer "pod".

Please ensure that this is the correct computer to license.

License request for interface eth0 (192.168.1.4) written to file
/ioncube_encoder/licreq-eth0.txt

Please email this license request file to licenses@ioncube.com
```

Windows

On Windows, the Encoder Installer creates a program group icon for generating license requests. Click the “Start Menu”, and in the ionCube program entry, select the “License Request Generator” item.

2.1.2 Submitting your License Request

Send the license request file for the interface you wish to license to licenses@ioncube.com. If files could not be created, please send the displayed license data instead. Be sure to include the `BEGIN` and `END` lines that you will see because these are part of the request. Your request will be processed, and if successful, a license will be allocated.

NOTE: If your computer has more than one network interface, you should only send a request for one interface, and the interface should always be active and visible to the operating system, but does not need to be connected to any network..

2.1.3 Specifying the License File

After processing your license request, you will be sent an email with details on creating your license file.

The Encoder expects to read license data from a file named `licdata.txt` in the same directory as the Encoder itself.

2.1.4 Transferring a License or Reinstalling a Machine

Although the Encoder is licensed on a per machine basis, it is possible to periodically transfer a license between machines, e.g. if upgrading. It is also recommended that this process also be performed if upgrading or reinstalling the operating system.

To transfer a license, a *revoke file* should first be generated either by using the `--revoke-license` Encoder command line option, or on Windows, with the Delete License item on the Windows Start/ionCube program group menu.

NOTE: The revoke must be performed on the machine that is *currently* licensed, and not on the new machine!

The revoke file will be created in the Encoder installation directory, and should be emailed to licenses@ioncube.com. Once the revoke file has been processed a new license request can be generated and a new license downloaded.

2.2 Running the Encoder

2.2.1 Command Line Format

The general form for running the Encoder to encode is either

```
ioncube_encoder [options] source -o target
```

or

```
ioncube_encoder [options] sources --into target
```

The `-o` option encodes `source` as `target`, where `source` and `target` are either both single files or directories. Using `--into` the Encoder sources can be one or more files or directories that are encoded into the target with the same name. See sections 2.3 and 3.1 for more details.

To syntax check use `-S` and specify one or more files or directories

```
ioncube_encoder [options] -S files_and_directories
```

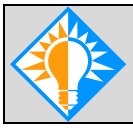
2.2.2 Passing Command Line Options

Most Encoder options use descriptive names, starting with `--`. Options requiring a value may use either an = character or white space to separate the option from its value.

Examples:

```
--add-comment=”Encoded by ionCube”

--add-comment “Encoded by ionCube”
```



Encoder options may be abbreviated to the shortest string that makes them unique. The Encoder will report an error if there are multiple choices.

2.2.3 Filename, Directory and Wildcard Pattern Matching

For options related to files, e.g. `--encode and --encrypt`, the Encoder provides flexible pattern processing to match files and directories by name or wildcard patterns.

Matching Files

Names with no trailing path separator are considered to be filenames.

Examples:

Encode all files with default extensions and any files named `x.inc`

```
--encode x.inc
```

Encode all files with default extensions and any files named `config/config.inc`

```
--encode config/config.inc
```

Encrypt all files in any directories named `templates`

```
--encrypt templates/
```

Directories

Appending a file separator specifies directories.

Example:

Ignore files in directory `config` and subdirectories

```
--ignore config/
```

Wildcard Matching

The Encoder also supports wildcard matching using the wildcard characters `*`, `?` and `[]`.

Wildcards are interpreted as follows:

Wildcard Character	*Matches*
`*`	Zero or more characters, e.g. `*.inc`
`?`	Any single character, e.g. `*.php?`
`[]`	Any character from a set, e.g. `*.php[34]`

Examples:

Encode files with default extensions and any files ending in `.inc`

```
--encode "*.inc"
```

Encode all files inside any directories named `config`

```
--encode "config/*"
```

Ignore all directories inside any directories named `config`

```
--ignore "config/*/"
```

Encode any files named `config` having one character after the name

```
--encode "config?"
```

Encode any files named `doc0`, `doc1`, `doc2`, … `doc7`

```
--encode "doc[0-7]"
```

Encode any files named `configx`, `configy` and `configz`

```
--encode "config[xyz].php"
```

2.2.4 Using Wildcard Characters on UNIX

When specifying an option value containing a wildcard character, be sure to use quotes or to escape the wildcard to prevent unintended shell expansion. Remember too that the Encoder can process entire directories. It is generally not necessary to use wildcards to select multiple files to be processed, and the containing directory can be named instead.

Examples:

Use

UNIX Version

```
--encode "*.ini"
```

```
--encode \*.ini
```

Instead of:

```
--encode *.ini
```

2.3 Quick-Start Encoding Examples

This section provides examples of how to use the Encoder to handle some common encoding requirements. Filenames and directory paths are examples only.

2.3.1 Encoding Single Files

Examples:

Files can be encoded as another by specifying the file as the source and using `-o` to name the target file. The target can be any filename.

Encode `/project/file1.php` as `/encoded-project/file1.php`

```
ioncube_encoder /project/file1.php -o /encoded-project/file1.php
```

Files can also be encoded *into* a directory using `--into` to name the target directory. The encoded file is will be given the same name as the source file.

Encode `/project/file1.php` into directory `/encoded-project`

```
ioncube_encoder /project/file1.php --into /encoded-project
```

More than one source item can be specified when using `--into` but giving a directory as the source is usually more convenient.

Encode `/project/file1.php` and `/extra/file2.php` into `/encoded-project`

```
ioncube_encoder /project/file1.php /extra/file2.php --into /encoded-project
```

2.3.2 Encoding Directories

Entire directory hierarchies can be recursively encoded by specifying a directory as the source. Files are either encoded or copied into the target. File attributes are also copied if possible, and on UNIX, any symbolic links will be replicated.

Examples:

Recursively encode `/project` as `/encoded-project`

```
ioncube_encoder /project -o /encoded-project
```

or

```
ioncube_encoder /project --into /encoded-projects
```

If the target directory already exists then you must specify how the Encoder should proceed. The available choices are:

Encoder Option	*Action*
`--replace-target`	Replace the target directory.
`--merge-target`	Merge files into the target directory.
`--rename-target`	Rename the existing target by appending a unique number.
`--update-target`	Similar to merge but only process the source file if its modified time is later than the target's modified time.

The `--replace` option is most commonly used.

2.3.3 Encoding Files with non-default File Extensions

By default the Encoder will encode files matching the pattern `*.php`, `*.php3`, `*.php4` or `*.phtml`, however files and directories matching any pattern or name can be encoded by using `--encode`. This option may be used as many times as required.

Example:

Encode files with default extensions and also any ending in `.inc`

```
ioncube_encoder --encode "*.inc" /project --into /encoded-projects
```

2.3.4 Encoding Shell Scripts

The Encoder will encode shell scripts which have PHP as their interpreter unless explicitly directed otherwise with `--copy` or `--ignore`. By default the first line of the source shell script will be copied to the target, but a custom line can be used with the following option

```
--shell-script-line '#!/usr/bin/php -q'
```

where the text inside the quotes is an example shell script line. Any encoded script can be used as a shell script by manually adding a shell script line to the encoded file after the encoding process is complete, and similarly an encoded PHP shell script will remain a valid encoded PHP script if the shell script line is removed. On UNIX it is important to enclose the option argument in single quotes as the '!' character has a special meaning for the shell.

2.3.5 Encrypting Templates and other Files

As well as protecting PHP files, the Encoder can encrypt any other files such as templates and images. Using the Loader API, encrypted files can then be decrypted only by an encoded file that was encoded by the same Encoder.

As an example of working with encrypted templates, a simple patch to the popular Smarty template engine is available on the ionCube website, allowing the Smarty engine to process both plain text and encrypted templates.

Example:

Encode PHP files with default extensions and encrypt files ending in `.tpl`

```
ioncube_encoder --encrypt "*.tpl" /project --into /encoded-projects
```

2.3.6 Leaving Files Unencoded

Files that would normally be encoded can be left unencoded and just copied by using `--copy`. This option may be used as many times as required.

Examples:

Encode `/project` into `/encoded-projects` leaving `config/config.inc.php` unencoded

```
ioncube_encoder --copy config/config.inc.php  /project --into /encoded-projects
```

Exclude all files matching `*_config.php` from being encoded

```
ioncube_encoder --copy "*_config.php"  /project --into /encoded-projects
```

Exclude files in `config` and subdirectories from being encoded

```
ioncube_encoder --copy config/ /project --into /encoded-projects
```

Exclude files in `config` from being encoded but not subdirectories

```
ioncube_encoder --copy "config/*" /project --into /encoded-projects
```

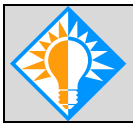
2.3.7 Omitting Files from the Encoding Target

When encoding a directory the Encoder will usually fully replicate the directory hierarchy. It can sometimes be necessary to ignore some items from the source tree and this is handled with the `--ignore` option. This option may be used as many times as required.

Example:

Ignore `RCS` files and backup files

```
ioncube_encoder --ignore RCS/ --ignore "*~" /project --into /encoded-projects
```



Use the `-v` option to check what files are being encoded, encrypted, copied, or ignored when using `--encode`, `--encrypt`, `--copy`, `--ignore` and `--keep`.

2.3.8 Adding Copyright and License Details to Encoded Files

Custom text can be added to the start of encoded PHP files, and this is useful for incorporating your own license or copyright messages. Files are protected so that any changes to an encoded file would prevent it from functioning, and so this text cannot be successfully removed.

Example:

```
ioncube_encoder --add-comment "Software written by FooSystems" --add-comment
        "Licensed to SomeCompany Inc." /project --into /encoded-projects
```

3 ENCODER COMMAND LINE OPTIONS

This chapter describes all available command line options grouped by their purpose.

3.1 Specifying the Source and Target

3.1.1 Source Items

The Encoder can be used to either encode single files or to recursively encode directories. Items to encode are passed to the Encoder without any associated option.

3.1.2 The Encoder Target [`-o`, `--into`]

The Encoder target may be specified in two ways. The `-o` option encodes a single source file or directory as a new name, and the `--into` option encodes one or more items into an existing directory.

Examples:

Encode file `hello.php` as `hello-encoded.php`

```
ioncube_encoder hello.php -o hello-encoded.php
```

Encode directory `/projects/test` as `/encoded-projects/test`

```
ioncube_encoder /projects/test -o /encoded-projects/test
```

Encode project `/projects/test` into `/encoded-projects`

```
ioncube_encoder /projects/test --into /encoded-projects
```

Encode projects `project1` and `project2` into `/encoded-projects`

```
ioncube_encoder /projects/project1 /projects/project2 --into /encoded-projects
```

Encode `file1.php` and `file2.php` into `/home/encoded-files`

```
ioncube_encoder file1.php file2.php --into /home/encoded-files
```

To protect against accidental error and possible loss of source files, the Encoder checks for being asked to encode directories that lie within the target tree or encoding into a directory that lies within the source tree. If encoding into the source tree is required then the option `--allow-encoding-into-source` may be used, see 3.15.2 below.

3.2 Encoded File Format

The Encoder can produce files in both binary and ASCII format. These formats are discussed below.

3.2.1 ASCII Format [`--ascii`]

By default, the Encoder produces encoded files in ASCII format. These files contain only printable characters, and may be safely transferred using FTP clients operating in either ASCII or binary mode without being corrupted.

3.2.2 Binary Format [`--binary`]

Binary format files are marginally more efficient than files produced with the default ASCII format. The advantages are slightly improved runtime performance and a smaller file size. A significant *disadvantage* is that some Windows programs in particular may corrupt binary encoded files during installation. Corruption can occur because the Windows operating system uses different characters than UNIX to represent line breaks, and some Windows archive and file transfer applications will change the line break characters when processing files. Examples are the CuteFTP file transfer program, WinZip if the *TAR smart cr/lf conversion* option is enabled (which it is by default), and FTP programs transferring in ASCII mode. Some PHP IDE's with FTP features only support ASCII file transfers, and so guarantee problems.

It may be advantageous to use Binary format files if installation of files is performed correctly, using tools that will not corrupt the files. However if this cannot be guaranteed, the default ASCII format is recommended as it still offers excellent performance, and will greatly reduce the chance of users accidentally corrupting files during installation.

3.3 Encoding to an Existing Directory Target

When an encoding target directory already exists, one of the following options must be used to tell the Encoder what you want it to do. There are four choices, to replace the target, update the target with changed files, merge files into the target or to rename the target.

3.3.1 Replacing the Target [`--replace-target`]

This option replaces an existing target, ensuring that the target contains no files from a previous encoding. This is the most common option to use when the target already exists.

3.3.2 Merging into the Target [`--merge-target`]

This will preserve the existing target and all files encoded or copied from the source will be created or overwrite any existing files. Note that any files already part of the target but not present in the source project are preserved.

3.3.3 Renaming the Target [`--rename-target`]

This will rename the target directory by appending a number to it. The number used will be the smallest number to produce a directory name that does not already exist. It is effectively a backup option.

3.3.4 Updating the Target [`--update-target`]

This option is similar to `--merge-target` except that files are only processed if the modified time of the source file is greater than the modified time of the old target file, or the target file does not exist.

3.4 Selecting Files to be Encoded, Encrypted, Copied or Ignored

By default, the Encoder will encode files having names ending in `.php`, `.php3`, `.php4` or `.phtml`, and encrypt any files specified with `--encrypt`. All other files will be copied. This section explains how to encode and encrypt files with other extensions, how to prevent files from being encoded, and how to exclude files from being part of the encoding target. For more examples, please see section 2.3. Note that you can use the options described here multiple times and in any order to describe precisely how you require the Encoder to process your project.

3.4.1 Encoding Specific PHP Files [`--encode`]

Use `--encode` to specify additional file patterns, files or directories to encode, or to reverse the effect of `--copy` (see 3.4.2 below).

Examples:

Encode all files ending in `.inc` as well as the default extensions

```
--encode "*.inc"
```

Also encode the file `licenses/license.key`

```
--encode licenses/license.key
```

Encode files in directory `tests/encoded`

```
--encode tests/encoded/
```

This last example would be useful if you had used `--copy tests` to tell the Encoder to copy the directory tests, but where you want the subdirectory `encoded` to be encoded. Note that this does not request *all* files to be encoded, but ensures that any files matching the default extensions or other file patterns will be.

Encode all files in directory `tests/encoded` but not subdirectories

```
--encode "tests/encoded/*"
```

3.4.2 Encrypting Files [`--encrypt`]

Use `--encrypt` to specify files or directories that are to be encrypted. Encrypted files can be decrypted by the Loader API `ioncube_read_file()` function, see section 5.4.1, and only when called from a file encoded by the same purchased Encoder installation. Typical examples would be encoding template or XML files.

Examples:

Encrypt files ending in `.tpl` or `.xml`

```
--encrypt "*.tpl" --encrypt "*.xml"
```

3.4.3 Excluding Files from being Encoded or Encrypted [--copy]

Use `--copy` to exclude files from being encoded or encrypted and to copy them to the target directory.

Examples:

Copy `user_config.php` instead of encoding it

```
--copy user_config.php
```

Copy files in directory `config` and subdirectories

```
--copy config/
```

Copy files in directory `config` but still encode files in any subdirectories

```
--copy "config/*"
```

3.4.4 Excluding Files from Target [--ignore]

Use `--ignore` specifies files and directories to ignore entirely and exclude from the target directory.

Example:

Ignore `RCS` directories and emacs backup files

```
--ignore RCS/ --ignore "*~"
```

3.4.5 Including Ignored Files [--keep]

The effect of `--ignore` can be reversed by using `--keep`.

Example:

Ignore all files in directory `docs` except for `README`

```
--ignore docs/ --keep docs/README
```

The Encoder applies the options above in the order that they appear. Combining them can achieve precise control over which files are to be encoded, encrypted, copied, or that will be excluded. The `-v` option is useful to see the effect of these options and will show how files were processed.

3.4.6 Including only Encoded Files into the Target [--only-include-encoded-files]

This option will produce a target containing only encoded files. Files that would otherwise be copied are ignored.

3.5 Bytecode Obfuscation

The ionCube PHP Encoder achieves a high level of protection by compiling PHP source code into PHP bytecode, storing this in a proprietary format, and executing code inside the Loader component. This is an analogous operation to compiling a C program into native machine code. Although the entire source code is eliminated during this process, as with compiled C programs, some symbols must remain. As a further level of protection, the ionCube Encoder can apply a *one-way binary transformation algorithm* to obfuscate certain strings, and elements such as function names will then not be exposed in their original form by PHP features such as `get_defined_functions()`. In addition to user controlled obfuscation, other obfuscations are also automatically applied during compilation and at runtime.

3.5.1 Obfuscating Compiled Bytecodes [`--obfsucate`]

The Encoder can obfuscate the names of global functions, the names of local variables in global functions, and line numbers. The easiest way to use this option is shown in the following example:

Obfuscate line numbers, function names and local variables

```
--obfuscate all
```

This option can also be used with any combination of the tokens `functions, linenos, locals,` to allow the obfuscation of any combination of functions, line numbers and local variables. Separate tokens with a comma and no whitespace.

Example:

Obfuscate line numbers and function names but not local variables

```
--obfuscate linenos,functions
```

3.5.2 Specifying an Obfuscation Key [`--obfuscation-key`]

Even though the obfuscation uses a one-way, non reversible algorithm, a custom obfuscation key prevents any chance of reversal by chance through observation of known obfuscations. The obfuscation key must be supplied, and the Encoder will generate an error if it is missing.

Example:

```
--obfuscate all  --obfuscation-key "the 5 claw red dragon"
```

3.5.3 Specifying Obfuscation Exclusions [`--obfuscation-exclusion-file`]

While it is desirable to obfuscate an application, it is sometimes necessary to prevent certain functions from being obfuscated. For example, functions external to the application and called by the application should not be obfuscated. Similarly, application interface functions called by external scripts should not be obfuscated.

To specify these exceptions, a text file should be created with the name of each function, one per line, and the option `--obfuscation-exclusion-file` should be used to pass the name of the file to the Encoder.

It is not necessary to add the names of built-in PHP functions to this file.

3.6 File Based Server Restrictions (Pro and Cerberus Editions)

The Pro and Cerberus Encoders can optionally encode files with server restrictions such that they will stop functioning beyond some point in time, and also to restrict the machines on which the files can be run. The options for restricting files are described below. As an alternative, files can be restricted to require a license file containing time and server restrictions. The License file approach is often more flexible, and is described in section 3.7 below.

3.6.1 Expiring Files after a Period [`--expire-in`]

Files can be set to expire after a given number of seconds, minutes, hours or days by using:

```
--expire-in <period>
```

where `<period>` is a number followed by either `s`, `m`, `h` or `d` to denote a period in seconds, minutes, hours or days.

Examples:

Expire files in 7 days

```
--expire-in 7d
```

Expire files in 8 hours

```
--expire-in 8h
```

Note that the expiry period is relative to the time that files are encoded.

3.6.2 Expiring Files from a Date [`--expire-on`]

Files can be set to expire from a specific date by using:

```
--expire-on <yyyy-mm-dd>
```

Example:

Expire files from 2007-01-01

```
--expire-on 2007-01-01
```

3.6.3 Locking Files to Specific Domains and Servers [`--allowed-server`]

Encoded files can be restricted to load only on machines with specific IP addresses and/or domain names. The domain name is also referred to as the server name[1]. Using the Cerberus version, encoded files can also be restricted by MAC (Ethernet) address.

The complete syntax for server restricting files is:

```
--allowed-server [<domain names>][@<IP addresses>][{<MAC address>}]
```

Each server specification can contain as many domain names and IP addresses as desired. If using Cerberus, a MAC address restriction may also be given. At least one type of restriction must be specified, but items are optional. The option may also be used more than once to specify multiple restrictions.

[1] The server name is an attribute set by the web server for each domain or virtual host, and is the value accessed in PHP as `$_SERVER['SERVER_NAME']`.

Restricting by Domain Name

Specify domain names separated by commas.

Examples:

Restrict files to `www.foo.com`

```
--allowed-server www.foo.com
```

Restrict files to `www.foo.com` or `www.bar.com`

```
--allowed-server www.foo.com,www.bar.com
```

Restrict files to server name `1.2.3.4`

```
--allowed-server 1.2.3.4@
```

Note the trailing @ after the domain name. A domain name will rarely have the same format as an IP address, but if it does then appending the @ tells the Encoder that the item is a domain name and not an IP address.

Restricting by IP Address

IP addresses may be specified as a single address, a range of addresses or a subnet. Multiple addresses should be specified separated by commas.

Examples:

Restrict files to `192.168.1.4`

```
--allowed-server 192.168.1.4
```

Restrict files to `192.168.1.4` and `192.168.1.20`

```
--allowed-server 192.168.1.4,192.168.1.20
```

Restrict files to `192.168.1.20` through `192.168.1.25`

```
--allowed-server 192.168.1.20-192.168.1.25
```

or

```
--allowed-server 192.168.1.20-25
```

Restrict files to `192.168.1` subnet

```
--allowed-server 192.168.1
```

Restrict files to subnet with 28 significant bits

```
--allowed-server 192.168.1.255/28
```

Restricting by MAC Address

MAC addresses are composed of 6 bytes and should be specified using hex notation as follows.

Example:

Restrict to MAC `00:01:02:06:DA:5B`

```
--allowed-server {00:01:02:06:DA:5B}
```

Combining Restrictions

Examples:

Restricting files to a domain name and an IP address

```
--allowed-server www.foo.com@192.168.1.2
```

Restricting files to a domain name and specific MAC address

```
--allowed-server www.foo.com{00:02:08:02:e0:c8}
```

Restricting to either of two domains on either of two IP addresses

```
--allowed-server www.foo.com,www.bar.com@192.168.1.1,192.168.1.3
```

Restricting to a domain name, IP address and MAC address

```
--allowed-server www.foo.com@192.168.1.1{00:02:08:02:e0:c8}
```

3.7 License Based Server Restrictions (Pro and Cerberus Editions)

A flexible alternative to file based restrictions, described above, are license based restrictions. With this mechanism, files are encoded to require a license file, and the license file contains time and server based restrictions. A major advantage compared to file based restrictions is that the encoded files may be encoded just once, with a license file containing custom restrictions created for each installation. This is especially beneficial when producing software updates as a single encoded update may be made available to all users, and existing license files will control the new encoded files. With file based restrictions, it would be necessary to produce an update encoded for each installation with the same restrictions as the original installation.

3.7.1 Specifying a License File [`--with-license`]

To restrict files to require a license file, use the option

```
--with-license path
```

to specify the path of the license file that should be used to restrict the execution of the encoded script. At runtime, the Loader will search for the license file relative to the location of the encoded script, so a relative path should be used when specifying the license. The path will often be just the name of the license file, e.g. `license.txt`

Typically an application will have a single top level directory. In this case the license file could be saved into this top level directory, and the filename of the license could be used on the command line instead of a more complicated relative path. If the Loader cannot locate the license file relative to the PHP script that needs it, the Loader will search parent directories until either the License file is found or there are no further parent directories.

3.7.2 Specifying a Passphrase [`--passphrase`]

License files are encrypted using industry proven algorithms, and with the encryption keyed with a passphrase. Use the command line option

```
--passphrase phrase
```

to specify a passphrase. The passphrase used when encoding files must match the passphrase used to generate the corresponding license file in order for the Loader to successfully decrypt the license file when the script is executed. See section 4.2.2 below for more details. If the Loader cannot decrypt the license file it will prevent execution of the script.

3.7.3 License Check Mode [`--license-check`]

When an encoded file restricted by a license is read by the Loader, there are two methods by which the license restrictions can be enforced. The Loader can automatically ensure that all server restrictions are matched, the license has not expired, and that all enforced properties are matched in the license file (see section 4.2.8 below). This is the default method, but it can also be specified by encoding with the option

```
--license-check auto
```

A script can also use the Loader API to validate properties, server restrictions, and any expiry date contained in the license. In order to implement a manual license check, and so prevent the Loader from automatically validating the license, encode files with the option

```
--license-check script
```

Several Loader API functions useful when implementing a manual license check are described in chapter 5. Please be careful to ensure code security when implementing a license validator in PHP. In particular if a validator is contained in a file which will be included in each top-level script, then it is necessary to use Include File Protection to ensure that a hostile user cannot replace the validator with a different file.

3.8 Target File Attributes

3.8.1 Copying with Hard Links [--use-hard-links]

The Encoder will normally copy any files into the target that are neither encoded nor encrypted. This is fast, but performance can be improved and disc space saved by using the `--use-hard-links` option to replicate by using hard links. This feature is only available with UNIX Encoders, and only if the source and target files are on the same filesystem.

3.8.2 Using Default File Permissions [--without-keeping-file-perms]

This option applies the default file permissions to target files instead of copying permissions of the corresponding source file.

3.8.3 Updating File Times [--without-keeping-file-times]

This option creates target files with a current timestamp instead of copying times from the corresponding source file.

3.8.4 File Ownership [--without-keeping-file-owner]

When running as root on UNIX the Encoder will usually copy file ownership from source files and directories. This option will create target items as the user running the Encoder.

3.8.5 Setting File Ownership [--apply-file-user, --apply-file-group]

When running the Encoder as root on UNIX, different user and group ID's can be set for target files using:

```
--apply-file-user <user id/name>

--apply-file-group <group id/name>
```

The `id` or `name` may be either a numeric id or a name.

3.9 Language Options

3.9.1 Ignoring Short Open Tags [`--no-short-open-tags`]

Use this option to only recognise PHP files that use `<?php ?>` style tags. By default the Encoder recognises both `<?php ?>` and `<? ?>` style.

3.9.2 Strict Language Usage [`--strict-php`]

Some features of the PHP language are now deprecated and may disappear from future versions of the language. By default the Encoder permits deprecated language features to be used, but this option will enable warnings when deprecated features are used.

3.10 Encoded File Header Customisation

Encoded files contain a PHP header (the *preamble*) that, if required, will perform the run-time installation of the Loader. It will also produce an error message if no Loader could be installed or in some cases of file corruption. The default header is ideal for most cases, however Encoder options support customising parts of the header and for setting an entirely new header.

Files may also have custom comments added. This feature offers the addition of plain text copyright messages, product version number, contact details, and so on. Embedded digital signatures protect encoded files from tampering, and any modifications to an encoded file will render it useless. Therefore such messages cannot be successfully changed or removed.

3.10.1 Removing Run-Time Loader Support [`--without-runtime-loader-support`]

This option shortens the header by removing support for run-time install of the Loader. This is useful if your encoded files will be installed on a system where you know that run-time installation of the Loader is not required. The header will still contain code to generate an error if no Loader is installed.

3.10.2 Generating Files with no PHP Header [`--without-loader-check`]

This option produces files with no PHP header and only the encoded file data. When running files without a header there will be no error produced if there is no Loader installed and the Loader must be installed in the `php.ini` file.

3.10.3 Customising the 'no Loader installed' Message [`--message-if-no-loader`]

To customise the message produced if no Loader is installed, use:

```
--message-if-no-loader <text>
```

`<text>` must be a valid PHP expression and is passed to the PHP `die()` function.

Example:

```
--message-if-no-loader "'No Loader is installed. Please contact support.'"
```

Note the use of single quotes around the message because a string is being passed to the `die()` function.

3.10.4 Customising the 'no Loader installed' Action [`--action-if-no-loader`]

To customise the action when no Loader is installed, use:

```
--action-if-no-loader <php code>
```

3.10.5 Setting the Run-Time Loader Path [--loader-path]

To change the run-time Loader path, use:

```
--loader-path <path>
```

The current default setting performs selection of the Loader based on operating system type and PHP version, and is:

```
'/ioncube/ioncube_loader_'.$__oc.'_'.substr(phpversion(),0,3).(($__oc=='win')?'.dll':'.so')
```

$__oc is predefined in the header as:

```
strtolower(substr(php_uname(),0,3))
```

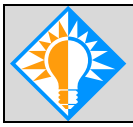
Changing the Loader path may be useful if you wish to distribute Loaders with your application but in a different directory, or if you wish to use run-time Loading but do not need to use the dynamic selection of the Loader.

3.10.6 Setting the Header Code [--preamble-file]

The entire PHP header may be set by using:

```
--preamble-file <file>
```

<file> should be the path to a file containing PHP code to place at the start of the encoded files.



Download the current PHP header as a starting point from either
www.ioncube.com/resources/rtl_php_header.tar.gz or
www.ioncube.com/resources/rtl_php_header.zip

3.10.7 Header Comments [--add-comment, --add-comments]

To add text to appear as comments at the start of encoded files, use:

```
--add-comment <text>
```

The option may be used as many times as required.

To add comments from a file, use:

```
--add-comments <file>
```

Examples:

```
--add-comment "Copyright FooBar Inc. 2003" --add-comment "All Rights Reserved"

--add-comments custom/comments.txt
```

3.11 Customising Loader Behaviour

3.11.1 Loader Event Messages [--loader-event]

Loader messages generated by run-time events can be customised at encoding time to those of your own choice.

For each web request, Loader messages customised by an encoded file will take effect for all other encoded files unless a later included file provides a different message.

To customise an event message, use:

```
--loader-event "<event>=<message>"
```

<event> should be the event type to customise and <message> the text to associate with the event.

Event types are:

Event Type	*Triggered When…*
corrupt-file	An encoded file has been corrupted.
expired-file	An encoded file has reached its expiry time.
no-permissions	An encoded file has a server restriction and is used on a non-authorised system.
clock-skew	An encoded file is used on a system where the clock is set more than 24 hours before the file was encoded.
untrusted-extension	An encoded file was encoded with the --disallow-untrusted-extensions option, see 3.12 below, and is used on a system with an unrecognised extension installed.
license-not-found	The license file required by an encoded script could not be found.
license-corrupt	The license file has been altered or the passphrase used to decrypt the license was incorrect.
license-expired	The license file has reached its expiry time.
license-property-invalid	A property marked as 'enforced' in the license file was not matched by a property contained in the encoded file.
license-header-invalid	The header block of the license file has been altered.
license-server-invalid	The license has a server restriction and is used on a non-authorised system.
unauth-including-file	The encoded file has been included by a file which is either unencoded or has incorrect properties.
unauth-included-file	The encoded file has included a file which is either unencoded or has incorrect properties.
unauth-append-prepend-file	The php.ini has either the --auto-append-file or --auto-prepend-file setting enabled.

Example:

```
--loader-event "expired-file=This software has expired."
```

Custom messages may also contain display formats. Each format is replaced with specific text as follows.

Format	*Replaced With*
`%f`	The path of the file generating the event.
`%i`	Server IP address ('no-permissions' event only).
`%h`	Server name ('no-permissions' event only).
`%n`	The path of the unauthorised including or included file.

3.11.2 Callback Files [`--callback-file`]

To implement a more elaborate error handling mechanism than with Loader Events, a *callback file* can be specified to handle Loader error cases. Use the option

```
--callback-file <relative-path>
```

to specify a callback file. The path should be relative to the top-level directory of the PHP application. In particular, if the callback file is contained in the top-level directory, then specify the filename rather than a full path.

The callback file should contain a function with the signature:

```
function ioncube_event_handler($err_code, $params)
```

The error code will be passed as the first argument, and an associative array of context-dependent values will be passed as the second argument. The error code is an integer, and for convenience the Loader defines constants for all event error codes. See section 3.11.3 for a list of all constants.

The name of the file that caused the error is always passed as a parameter with key `current_file`, and for server restriction errors parameters are passed with keys `domain_name` and `ip_address`. The path to the expected license file is passed with key `license_file`, and for errors related to include file restrictions, the file that included the encoded file, or was included by the encoded file, is passed with key `include_file`.

3.11.3 Loader Event Constants

The Loader defines PHP constants corresponding to the supported error codes. These codes correspond to the Loader events described in the previous section follow:

Value	*PHP constant*	*Loader Event*
1	ION_CORRUPT_FILE	corrupt-file
2	ION_EXPIRED_FILE	expired-file
3	ION_NO_PERMISSIONS	no-permissions
4	ION_CLOCK_SKEW	clock-skew
5	ION_UNTRUSTED_EXTENSION	untrusted-extension
6	ION_LICENSE_NOT_FOUND	license-not-found
7	ION_LICENSE_CORRUPT	license-corrupt
8	ION_LICENSE_EXPIRED	license-expired
9	ION_LICENSE_PROPERTY_INVALID	license-property-invalid
10	ION_LICENSE_HEADER_INVALID	license-header-invalid
11	ION_LICENSE_SERVER_INVALID	license-server-invalid
12	ION_UNAUTH_INCLUDING_FILE	unauth-including-file
13	ION_UNAUTH_INCLUDED_FILE	unauth-included-file
14	ION_UNAUTH_APPEND_PREPEND_FILE	unauth-append-prepend-file

3.12 Encoded Files and Untrusted/Open Source Extensions

Even though the ionCube Loader has many security features that make hacking or reverse engineering efforts time consuming and almost certainly unsuccessful, you may wish to allow your files to run only if "trusted" extensions are installed. Trusted extensions are closed-source extensions that we know to be genuine. Open Source extensions are untrusted because by their very nature, they can be modified to behave in ways that were not intended by their well-meaning creators.

3.12.1 Blocking Untrusted Extensions [`--disallow-untrusted-extensions`]

Use the above option to not allow your files to work if untrusted engine extensions are installed.

3.13 File Properties and Include Attack Prevention

File *properties* are key-value pair data items that are securely stored as metadata in encoded files, separate to the PHP code. Properties can be read by a Loader API function, and also used with the *include attack* prevention system. Include attacks are where program scripts are replaced by unauthorised ones in an attempt to change program behaviour. To guard against this, encoded files can be protected so that they can only be included by a file with specific properties defined, and so that they can only include a file if it has certain properties. This powerful feature can also help prevent unauthorised use of libraries by allowing your included files to only be included by your own application, and not by someone else's program.

3.13.1 Setting Properties [`--property, --properties`]

To define one or more properties, use:

```
--property "name[=value]"

--properties "name[=value][, ...]"
```

`name` is the name of the property to be defined and `value` is the property value. Use a comma to separate multiple properties.

Values can be numeric, a string that is optionally delimited by `' '` or `" "`, or an array delimited by `{ }`. Array elements may optionally have keys.

Examples:

Define a property `version` with integer value `5`

```
--property version=5
```

Define a property `version` with string value `"2.0"`

```
--property "version='2.0'"
```

Define several properties

```
--properties "version=1,company='Foo Technologies'"
```

Define an array

```
--property "features={options=>{'save','load'},licensed_to=>'Foo Tech Inc.'}"
```

3.13.2 Include Attack Prevention [--include-if-property]

To restrict which files can be included by a file and also the files that can include a file, use:

```
--include-if-property "name=[value][, ...]"
```

Property name and value are as defined in section 3.13.1 above.

This option may be used more than once to define multiple sets of required properties.

Examples:

Include files if property program_name has value "my app"

```
--include-if-property "program_name='my app'"
```

Include files if property pname is either "app1" or "app2"

```
--include-if-property "pname='app1'" --include-if-property "pname='app2'"
```

3.13.3 Preventing Prepend and Append File Usage [--disable-auto-prepend-append]

By utilising the auto_prepend_file and auto_append_file php.ini settings it is possible to specify a PHP file which should run before or after any other scripts. This may undermine the security of encoded PHP scripts, and can be disabled using the option

```
--disable-auto-prepend-append
```

If this option is used and a server has either the append or prepend php.ini setting enabled, the encoded scripts will not run. Some servers may have a legitimate reason for enabling these settings, so this Encoder option is not enabled by default.

3.14 Project Handling

Setting up the Encoder for single-command repeat encoding of projects can easily be performed using UNIX shell scripts or Windows batch files, however the Encoder also has a built-in projects handling feature that may usefully be used as well or instead.

The projects feature uses a project file to store and provide command line options. Project file options are merged with any additional options passed to the Encoder, and the project file may be updated or recreated when required. As the project file is a plain text file, it can also be edited if necessary.

3.14.1 Specifying the Project File [`--project-file`]

The project file to use is set with:

```
--project-file <file>
```

Once a project file has been created, to encode with the given project options only this option need be used.

3.14.2 Creating the Project File [`--create-project`]

This option creates the project file named with `--project-file`. The file is created or overwritten, and is set with whatever other options are used to the Encoder.

Examples:

Create and initialise project file `p1`

```
ioncube_encoder --project-file p1 --create-project /project1 -into /encoded-apps
```

Repeat encoding based on project file `p1`

```
ioncube_encoder --project-file p1
```

Repeat encoding based on project file `p1` but with verbose mode enabled

```
ioncube_encoder --project-file p1 --verbose
```

3.14.3 Update a Project File [`--update-project`]

This option updates a project file by merging in any new options.

Example:

Repeat encoding based on project file `p1` but permanently add the `--replace` option

```
ioncube_encoder --project-file p1 --replace -update-project
```

3.15 Miscellany

3.15.1 Encoding and Bytecode Optimisation [--optimise, --optimize]

By default, the Encoder uses an encoding format that encodes with the best Encoder performance and good run-time performance. At the expense of increased encoding time, smaller files with possibly better run-time performance may be obtained by increasing the optimisation level.

The options:

```
--optimise more

--optimise max
```

increase optimisation to either an intermediate or a maximum level.

The option `--optimize` is an alias for `--optimise`

3.15.2 Allowing Encoding into the Source Tree [--allow-encoding-into-source]

By default, the Encoder prevents a target directory to be within the source tree or for the source directory to be within the target tree. This is to prevent accidental overwriting of source files or unexpected results. The `--allow-encoding-into-source` option allows this. To avoid the target being treated as part of the source tree use the `--ignore` option to ignore the target.

Examples:

In these examples we have a simple project in a directory called `test`. The project contains the file `helloworld.php`

Encoding with the default safety check

```
$ ioncube_encoder test -o test/encoded

Error: Can't encode to a directory within the source tree
```

The Encoder safety check prevented encoding because the target is within the source tree.

Encoding with the default safety check disabled

```
$ ioncube_encoder test -o test/encoded --allow-encoding-into-source --ignore encoded/

$ ls -R test

test:

encoded/  helloworld.php

test/encoded:

helloworld.php
```

Source directory with source file and encoded target

Target directory with encoded result

3.15.3 Documentation Comments [--no-doc-comments]

Documentation comments are comments with the following syntax:

```
/**
My code comment
*/
```

These comments are exposed by the PHP 5 reflection API, and are preserved by the PHP 5 Encoder by default. In order to omit these documentation comments from encoded files specify the `--no-doc-comments` option.

3.15.4 Enforce Minimum Loader Version [--min-loader-version]

This option encodes files with a requirement that the Loader version is of the specified version or greater. This can be useful to ensure that a particular feature of the Loader is supported. The option should be specified as a version string in the format `major[.minor[.revision]]`.

Examples:

```
--min-loader-version 3.1

--min-loader-version 3.1.5
```

3.15.5 Program Version [-V, --version]

To display the program version, use:

```
-V or --version
```

3.15.6 Verbose Mode [-v, --verbose]

Verbose mode will produce details of Encoder operations and progress.

3.15.7 File Verify [--verify]

If run-time loading is to be used then files must be able to be read and parsed by the PHP engine as valid PHP files. This will increase encoding time, and is a legacy option that would only be necessary if you had customised the PHP header and wish to check it.

3.15.8 Help [--help]

This option displays a summary of Encoder options.

4 LICENSE FILE GENERATION (Pro and Cerberus Editions)

4.1 Introduction to License Files

The Pro and Cerberus editions of the PHP 4 & 5 Encoder bundle offer powerful license file creation features, with restrictions that can be enforced automatically by the ionCube Loader at runtime, or by your own scripts and using functions in the Loader API. A license generator program is also included with the PHP 4 & 5 Bundle product for generating license files.

Scripts can be restricted to run only in the presence of a license file, properties can be set in the license file that must match properties set in the encoded files, and license files can be used to restrict encoded files to a particular machine. Licenses can also be set to expire at some point in the future.

Benefits of License Files

A benefit of using license files as an alternative to setting restrictions to the encoded files themselves, is that projects will not need to be encoded every time in order to create a custom build for each end-user. This is particularly helpful when producing software updates to existing customers, as with license files, one encoded update will work for all installations. If restrictions are instead associated with each encoded file, a product update would need to be encoded for each existing customer, complicating the process of issuing a product update.

Server Restrictions

License file server restrictions can be supplied to the license generator in one of two ways. If the details of the server to be licensed are known, such as IP address, then these can be used explicitly. Often, however, details are unknown, and the goal is simply to restrict to a server without knowing the actual parameters. To support this, a Loader API function can be used to generate server data containing information about the target server, and after being received via a method such as email or a web form, the data can be passed to the license generator to create a license file for that server. This can be ideal for automating license generation based on information supplied from the installed PHP scripts during a licensing procedure.

Custom Properties

In addition to restrictions, custom key value properties can be added to license files and read via the Loader API. This feature might be used to customise product behaviour based on information read from the license file.

File Format

A license file consists of a header in plain text, followed by an encrypted data block. Any properties or restrictions in the license data can be optionally exposed, e.g. the expiry time, and additional text can be added to the header if desired. Finally, license files are protected by signatures to prevent removal or changing of the plain text.

Locating License Files

Each encoded file contains a relative path to its associated license file. Typically this will be just the filename, e.g. license.txt. When accessing a license protected encoded file, the Loader first looks for the license file path relative to the same directory as the encoded script. If not found, parent directories will be searched until the license file is found or until the directory root is reached. This allows for easy installation and management of license files for both shared and dedicated servers.

4.2 Creating License Files

4.2.1 Command Line Usage

The general form for running the command line license generation tool is

```
make_license --passphrase phrase -o output-path
```

When encoding files that require a license it is necessary to specify a product passphrase. The passphrase specified when encoding a product should agree with the passphrase specified when generating the corresponding license, and should generally be unique for a particular product. The output path is the path to which the new license file will be saved.

4.2.2 Using Passphrases to Differentiate Products [`--passphrase`]

As mentioned previously, it is recommended that a unique passphrase be used for each product or product variant that is encoded. This ensures that a license for one product will not unlock a second product. As an additional layer of security, a license created with one Encoder installation cannot unlock files encoded with a different Encoder installation, even if the passphrases were the same. The option `--passphrase` should be used to specify the passphrase.

4.2.3 Setting License Restrictions Explicitly [`--allowed-server`]

The `--allowed-server` option allows explicit setting of a license file restriction when target server details such as domain name or IP address are already known. The specification syntax is the same as for the similar option of the Encoder. Please see section 3.6.3 above for details of the syntax and examples.

4.2.4 Setting License Restrictions from Server Data [`--use-server-file`]

An alternative to setting license restrictions explicitly is to use server data collected by the application being licensed from the target server. Server data is obtained by calling the Loader API function `ioncube_server_data()`, and would then be passed back to be used for licensing. See section 5.2.3 for more details on this API function.

Once server data has been received, e.g. in an email or through a web form, and then written to a temporary file, the `--use-server-file` option can be used as follows

```
--use-server-file path
```

where *path* refers to the location of the server data file. As the machine may contain multiple network adaptors, it is also necessary to use either the option `--select-server-adapter` or `--select-adapters` to select which details to license to. These options are described in the next section.

4.2.5 Selecting Adapters [`--select-server-adapter, --select-adapters`]

Server data generated using the `ioncube_server_data()` API function includes all network interfaces. In addition, if the API function is called from a script via the web server, both the domain name and server IP address for the request will be stored if that information was available. This is the usual case, and it will generally be desirable to license to the information associated with the web request as this would usually be the same when accessing the main parts of the licensed product. In other cases, licensing to one or more adapters explicitly may be preferred. Both of these cases are catered for.

To license to the server information associated with the web request that called the server data API function, use the option

```
--select-server-adapter
```

Provided that the server data was requested from a page under the same domain as the application to be licensed, errors from mistaken domain names or IP addresses should not arise. When using this option, if no IP address or domain name was reported then the `make_license` program will exit with code 2 to allow handling of this case.

To license to one or more specific adapters, the option

```
--select-adapters <adapter list>
```

can be used to select the adapters. Here *adapter list* is either a comma separated list of numbers that refer to the position of the adapter in the server data file, or the `*` character. The first adapter is identified as `1`, and using `*`, all adapters will be licensed.

4.2.6 License Expiry [`--expire-in, --expire-on, --expose-expiry`]

These options support the creation of time expiring licenses. Expiry information is stored in the encoded part of the license file, but can also be exposed in the header block by using the `--expose-expiry` option.

These options take arguments in the same way as the options in sections 3.6.1 and 3.6.2 for setting expiry time on files, and examples are included in those sections.

4.2.7 License Properties [`--property, --expose-property`]

Custom key/value pair property data can be stored in a license file in the same way that properties can be stored in encoded files. Use the option syntax

```
--property "name[=value]"
```

to specify a property. Multiple properties can be specified in this way. See section 3.13.1 for more details on the supported syntax. Properties can also be exposed in plain text in the license header block by using the option

```
--expose-property name
```

4.2.8 License Property Checking [`--enforce-property`]

Properties may be included in a license either as a convenient mechanism for securely accessing custom data from an encoded script, or in order to lock a license to an encoded file. If a property is to be used for the latter purpose, the option

```
--enforce-property name
```

should be used. By default, encoded files secured by such a license must have a property with a matching key and value. If the property is not found then the Loader will exit before execution of the script begins. See section 3.7.3 for details on how to customise this behaviour.

4.2.9 Customising the Header Block [`--header-line`]

The text which occurs before the encrypted license data is called the *header block*. As with encoded files, this is protected from tampering and so it is important that this text is not edited after the license has been generated otherwise the license will become corrupted as a result. The header block content is determined by those properties which have been exposed, whether there is an exposed expiry date, and any custom header lines. To add custom lines to the header block, for each line use the command line option

```
--header-line text
```

4.2.10 Viewing Server Data Files [`--decode-server-file`]

Once saved to a file, the contents of data generated by `ioncube_server_data()` can be viewed with the `make_licence` program option

```
--decode-server-file path
```

where `path` is the path to a file containing server data. The domain name and server IP address that were reported by the web server for the request calling the API function are output first, followed by the name, IP address, and MAC address of each adapter installed on the server. To be both human readable and easily parsed, each line has a field name and value separated by a `:` character. If there was no domain name or IP address stored for the request, the field value will be the token `none`.

4.2.11 Troubleshooting License Problems

If an encoded script that requires a license fails, the particular error message displayed can give a clue as to the cause of the issue. For security reasons the error messages are general rather than specific.

If the license is reported to be *invalid* then a license property set in the license has not been matched in the encoded file. If the license is *corrupt* then either the contents of the license file have been altered, or the passphrase in the encoded file does not match the passphrase used to generate the license file. If the license is *not valid for this server* then a server restriction in the license has not been met.

If it is necessary to determine the contents of a license file the Loader API can be used. Encode a script with the options

```
--with-license license.txt --license-check auto
```

then use the Loader API to output the server restrictions, expiry date, and any properties contained in the license.

5 LOADER API

The ionCube Loader contains an API providing various functions and constants that may be useful to PHP scripts. Most functions return results dependent on whether or not the calling script was encoded.

5.1 File Information and Execution

5.1.1 Checking for an Encoded File [ioncube_file_is_encoded]

This function returns TRUE if the file containing the function call is encoded, and FALSE otherwise.

5.1.2 General Encoded File Information [ioncube_file_info]

This function returns FALSE if the file is not encoded. Otherwise it returns an associative array. The contents of the array are as follows:

Key	*Value*
FILE_EXPIRY	Either the file expiry time, or the license expiry time if a license file is present. The time is an integer in UNIX timestamp format: the number of seconds elapsed since midnight (00:00:00), January 1, 1970.
ENCODING_TIME	UNIX timestamp representing the time the file was encoded.
DEMO	TRUE if the file was encoded with an evaluation Encoder, otherwise FALSE.

5.1.3 Retrieving Properties Stored in an Encoded File [ioncube_file_properties]

This function returns an associative array consisting of file properties that were added to the encoded file with the --property or --properties command line option to the Encoder. Only properties defined in the calling script are returned.

5.1.4 Retrieving the Loader String Version [ioncube_loader_version]

This function returns the Loader version as a string.

5.1.5 Retrieving the Loader Integer Version [ioncube_loader_iversion]

This function returns the Loader version as an integer.

5.2 License and Server Information

5.2.1 Retrieving Properties Stored in a License [ioncube_license_properties]

This function returns an associative array consisting of license properties. Properties are added to a license by specifying the `--property` command line option to the `make_license` program. Each value in the associative array retrieved by this API function is itself an array with two values: the license property value itself, and a boolean value signifies whether the property is enforced.

Recall that an *enforced* property is one that the Loader will attempt to match with an encoded file property if the `--license-check auto` option is passed to the Encoder on the command line.

The return value of this function is `FALSE` if the calling file is not encoded or has no license file.

5.2.2 Retrieving the List of Permissioned Servers [ioncube_licensed_servers]

This function returns an array of server restriction specifications. These are the same strings which were specified on the command line when the license was created.

5.2.3 Creating a Server Data Block [ioncube_server_data]

When generating a license for an end user it will usually be necessary to retrieve information about the end user's server. This Loader API function generates a *server data block* containing information about the network adapters installed on the server and the server's domain name. This data block can then be used in conjunction with the `make_license` program to generate a license restricted to the user's domain and server.

This function can be called from either an encoded or unencoded script.

5.3 License Validation

5.3.1 Validating License Properties [ioncube_check_license_properties]

This API function returns TRUE if all enforced license properties are matched in the encoded file. Otherwise an array is returned consisting of all unmatched enforced properties.

5.3.2 Validating Licensed Servers [ioncube_license_matches_server]

This function returns FALSE if the file is encoded, requires a license, and the license has a server restriction which is not met by the current server. In all other cases the function returns TRUE.

In the case that an encoded script requires a license, but the license could not be found, the Loader will prevent execution of the script. This case does not occur, therefore, when calling the ioncube_license_matches_server() API function.

5.3.3 Validating License Expiry [ioncube_license_has_expired]

This function returns TRUE if the file is encoded, has a license, and the license has an expiry time which has passed. In all other cases the function returns FALSE.

5.4 Encrypted File Support

5.4.1 Reading Encrypted Files [ioncube_read_file]

```
mixed ioncube_read_file(string path
            [,bool &was_encrypted [,string passphrase] ] ] )
```

This API function can be used to read files encrypted by the Encoder with the `--encrypt` command-line option. If the file is read successfully the contents are returned as a binary-safe string. An integer is returned in the case of an error condition (use the PHP function `is_int()` to distinguish these cases). See 5.5 below for a table of error codes.

Both plain text and encrypted files can be read by this function, allowing the function to be used in situations where it is not known ahead of time whether a file will be encrypted. For example, a template engine could be designed that would accept both encrypted and unencrypted template files. If it is necessary to know whether a file was encrypted after it has been read the second optional argument (passed by reference) can be examined.

If an encrypted file has been written with a custom passphrase (i.e. a nonempty passphrase argument was passed to the `ioncube_write_file()` API function), the same passphrase should be specified as the third argument.

Files encrypted by one Encoder can only be read by PHP scripts encoded by the same Encoder, and encrypted files cannot be read by unencoded scripts.

5.4.2 Writing Encrypted Files [ioncube_write_file]

```
integer ioncube_write_file(string path, string content
            [,bool encrypt [,string passphrase] ] ] )
```

Encoded PHP scripts can write encrypted files using this API function. Files written in this way can be read with the `ioncube_read_file()` API function.

The first argument is the path of the output file.

The second argument is a binary-safe string containing the content to encrypt.

The optional third argument can be set to `FALSE` to write a plain text file.

The optional fourth argument can be used to specify a custom passphrase which will replace the default installation-specific passphrase. If a custom passphrase is used then files encrypted with one installation can be read by a different installation's encoded files, if the correct custom passphrase is passed to the `ioncube_read_file()` API function.

5.5 Error codes

The following table lists the error codes that can be returned from the API functions detailed in this section.

Code	*Meaning*
0	The file was successfully written.
1	The file could not be opened.
2	The file is corrupt.
3	An updated Loader should be installed to read the file.
4	An error occurred while reading the source file.
5	An error occurred while writing an encrypted file.
6	An error occurred while encrypting the file contents.
7	An encrypted file cannot be read by an unencoded PHP script.
8	The wrong passphrase was specified, or the wrong Encoder installation was used to encrypt the file.

6 ERROR REPORTING

The Encoder reports syntax errors in the emacs/xemacs style format of

```
filename:line number:message
```

This offers possible integration with emacs/xemacs and direct access to the point of error in source files. For example, given a directory of PHP files called `myproject`, running the xemacs `compile` command and specifying the compiler as

```
ioncube_encoder -S myproject
```

will syntax check all PHP files and report errors in a buffer. With the default xemacs key bindings, simply hitting `ctrl-X` will visit each file reported as containing an error and place the cursor at the line containing the error.

7 TROUBLESHOOTING

7.1 Unable to Start the Encoder

7.1.1 On UNIX

On UNIX if you receive an error similar to

```
$ ioncube_encoder

bash: ioncube_encoder: command not found
```

this is because the directory where the Encoder is installed is not listed in the shell `PATH` variable. It is recommended to use either a relative or absolute path to the Encoder rather than adding the Encoder directory to the `PATH` as this may cause product license related problems on some systems.

Examples:

Run from the current directory

```
./ioncube_encoder
```

Run the Encoder installed in `/usr/local/ioncube`

```
/usr/local/ioncube/ioncube_encoder
```

7.1.2 On Windows

On Windows, you can edit the `PATH` environment variable for the logged in user by going to the control panel, selecting the System item and clicking on the Environment tab.