ionCube PHP Encoder 6

Evaluation User Guide

ionCube Ltd. http://www.ioncube.com

CONTACTS & 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
Support: support@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
Support: support@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
Support: support@ioncube.pl

FAQ

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

Purchasing Products

To purchase ionCube products please visit http://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 encoding solution that delivers excellent security and run-time performance for PHP scripts. It is ideally suited for protecting projects of any size, and includes features to handle the most complex of requirements.

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 then applies several layers of encoding and transformations to produce the final platform independent files. This approach offers several advantages over other techniques, including superior protection from files never being restored to PHP source[1], and excellent run-time performance[2]. Encoder features allow for easy addition of plain text to the start of files, ideal for including custom copyright or license details. Encoded files also have embedded signatures to protect against tampering.

Encoded files are processed at run-time by the *ionCube Loader*. This component securely handles both the reading and execution of encoded files, and offers extra security over other techniques because the Open Source execution routine inside PHP is not used. The Loader requires no changes to the installed PHP software, and on most systems, the encoded files can automatically install the correct Loader when required. This powerful technique of *run-time loading* requires no changes to server configuration or server restarts, and is ideal for use on shared servers. Alternatively, permanent installation of the Loader is simply a one-line change to the `php.ini` file. The Loader is also compatible with Zend Optimiser and other popular extensions. To assist with installation, installation tools are also provided.

[1] Protection systems using source obfuscation or encoding of source are inherently insecure because files are restored to source at run-time. The source is then easily accessible inside the Open Source PHP engine. Files produced by the ionCube Encoder have no such problems as source files are turned into binary machine code data before encoding, and at run-time are restored only to binary code ready for execution.

[2] The usual time consuming PHP steps of parsing and compiling source are eliminated when running encoded files because this happens when files are encoded.

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 trial ware or customised versions of your software.

Key features include:

- Choice of binary or ASCII encoding formats.
- Recognition of `<?php`, `<?` and `<% %>` tags.
- Automatic syntax checking and error reporting of all encoded files.
- Recursive directory encoding with precise file and attribute replication.
- Easy control over the files and directories to be encoded, kept unencoded or ignored based on name or wildcard patterns.
- Adding user defined text to such as copyright or license details to encoded files.
- Optional *include-attack* protection to restrict which files can include encoded files.
- Associating key/value *properties* with files.
- File optimisation control.
- Projects feature to encode projects with pre-configured options.
- Customisation of run-time Loader messages.
- Full customisation of the run-time Loader install feature.
- Syntax-check only operation.
- Exceptionally fast encoding performance.

Features of the Pro Encoder include:

- All features of the basic Encoder.
- Generating files to expire after a time period or on a specific date.
- Generating files restricted by IP addresses and/or server names.

Features of the Cerberus Encoder include:

- All features of the Pro Encoder.
- Generating files restricted to Ethernet (MAC) addresses.

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 6.0
```

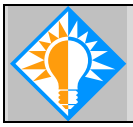
Shell input is shown in **`bold`**, and program output is `plain`. **`$`** is an example shell prompt for command entry but this may be different on your own system.

1.2.2 PHP 4 and 5 Encoders

In this document we shall refer to the Encoder command line executable as `ioncube_encoder`. This executable is the PHP 4 Encoder which should be used to maximise PHP script compatibility. The executable `ioncube_encoder5` is the PHP 5 Encoder which should be used when scripts contain language features only supported in PHP 5. Scripts encoded with the PHP 4 Encoder will run on both PHP 4 and PHP 5 servers.

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. The full features are described in chapter 3.

2.1 The Evaluation Encoder

2.1.1 Differences in the Evaluation Version

The Evaluation version includes all features of the Cerberus Encoder, and produces encoded files that are fully functional. However, encoded files will expire after 36 hours, although they can be re-encoded, and encoded files contain text indicating that they were produced by the evaluation version. This does not affect operation of the files, and the text is not present in files produced by the non-evaluation versions.

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 section 2.3 and 3.1 for more details.

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

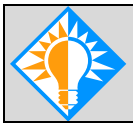
```
ioncube_encoder [options] -S files_and_directories
```

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 warn if there are multiple choices.

2.2.3 Filename, Directory and Wildcard Pattern Matching

For options related to files, e.g. `--encode`, 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
```

Copy all files, without encoding, in any directories named `templates`

```
--copy templates/
```

Directories

Appending a file separator specifies directories.

Example:

Ignore files in directory `config` and subdirectories

```
--ignore config/
```

Wildcard Matching

The Encoder 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 shell expansion.

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 `--include`. This option may be used as many times as required.

Example:

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

```
ioncube_encoder --include "*.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 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
```

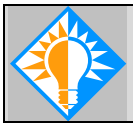
2.3.6 Omitting Files from the Encoding Target

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, copied, or ignored when using `--encode`, `--copy`, `--ignore` and `--keep`.

2.3.7 Adding Copyright and License Details to Encoded Files

Custom text can be added to the start of PHP files and is useful for incorporating your own license or copyright messages. Signatures protect the text so that any changes invalidate the

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 item as a new name and the `--into` option encodes one or more items into an existing directory.

Examples:

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

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

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

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

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

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

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

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

Encoding `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 3.13.2 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 more efficient than the default ASCII format files. The advantages are slightly improved runtime performance and a smaller file size. A disadvantage is that some Windows programs may corrupt binary encoded files during installation. Corruption can occur because the Windows operating system uses different characters to represent line breaks in files than UNIX, 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.

It is can 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 three choices – to replace the target, 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, not-Encoded or Ignored

By default, the Encoder will encode files having names ending in `.php`, `.php3`, `.php4` or `.phtml`. All other files will be copied. This section explains how to encode 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 sections 2.3.3 (Encoding Files with non-default File Extensions), 2.3.5 (Leaving Files Unencoded), and 2.3.6 (Omitting Files from the Encoding Target) above. Note that you can use the options described here multiple times, letting you tell the Encoder precisely how you would like it to process your project.

3.4.1 Encoding Specific 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 Excluding Files from being Encoded [`--copy`]

Use `--copy` to exclude files from being encoded 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/*"
```

3.4.3 Excluding Files from Target [--ignore]

--ignore specifies files and directories to ignore entirely and exclude from the encoding target.

Examples:

Ignore RCS directories and emacs backup files

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

3.4.4 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, not encoded, that appear in the target or that will be excluded. The -v option is useful to see the effect of these options and will show what files are being encoded, copied or ignored.

3.4.5 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 Setting Server Restrictions (Pro & Cerberus Versions)

3.5.1 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.

3.5.2 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 2004-01-01

```
--expire-on 2004-01-01
```

3.5.3 Locking Files to Specific Servers [--allowed-server]

Encoded files can be restricted to load only on machines with specific IP addresses and/or server names[3]. Using the Cerberus version, encoded files can also be restricted by MAC (Ethernet) address.

The complete syntax for server restricting files is:

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

Each server specification can contain as many server 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 be used more than once to specify multiple restrictions.

[3] The server name is an attribute set by the web server for each domain or virtual host.

Restricting by Server Name

Specify server 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 server name. A server 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 server 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 server name on an IP address

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

Restricting files to a server 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 server name, IP address and MAC address

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

3.6 Target File Attributes

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

The Encoder will normally perform a file copy into the target for non-encoded files. This is fast but performance can be improved and disc space saved by using the `--use-hard-links` option to copy 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.

3.6.2 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.

3.6.3 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.

3.6.4 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.

3.6.5 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.

3.7 Language Options

3.7.1 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.

3.7.2 Allowing ASP Tags [--allow-asp-tags]

This option allows PHP scripts having the Microsoft ASP open tag syntax of <% to be encoded.

3.7.3 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 warn when deprecated features are used.

3.8 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 provide for 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.

3.8.1 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.

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

This option produces files with no PHP header. 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.

3.8.3 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.

3.8.4 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>
```

3.8.5 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.

3.8.6 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
http://www.ioncube.com/rtl_php_header.tar.gz or
http://www.ioncube.com/rtl_php_header.zip

3.8.7 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
```

3.9 Customising Loader Behaviour

3.9.1 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 without the --allow-untrusted-extensions (see 3.10 below) option and is used on a system with an unrecognised extension installed
license-not-found	The license required by an encoded file could not be found.
license-corrupt	The license has been altered or the passphrase used to decrypt the license was incorrect.
license-expired	The license 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.

3.9.2 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 prototype

```
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, but the Loader defines constants for all the error codes. See section 3.9.3 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`.

3.9.3 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

3.10 Encoded Files and Untrusted/Open Source Extensions

Even though the ionCube Loader has many security features that make any hacking or reverse engineering efforts exceptionally 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.

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

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

3.11 File Properties and Include Attack Prevention

Encoded files can have key-value pair *properties* associated with them. Properties can be read by a Loader API function, and files can be restricted so that they can only be included by another file if that file has specific properties defined. This powerful feature helps in include attack prevention by allowing your included files to only be included by your own application, and not by someone else's program.

3.11.1 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.'}"
```

3.11.2 Restricting Include Access by Property Value [--include-if-property]

Files may be restricted to only allow inclusion if the including file has specific properties set and with particular values.

To restrict access, use:

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

Property name and value are as defined in section 3.11.1 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'"
```

Note that files having an include restriction will not be loaded if accessed directly. Therefore, any files needing to be accessed directly and not included must not have a property restriction in effect. To achieve this, a project may need to be encoded in two parts - first to encode files that should only be included by other encoded files and then to encode the top-level scripts.

Example:

To encode project app such that files ending in .inc can only be included by files that have property key with value 12345, the following steps could be followed.

1) Encode all files, define the property named key and restrict all files to requiring key to be defined.

```
ioncube_encoder app --into encoded-projects --encode "*.inc" --include-if key=12345 -
        -property key=12345
```

2) Re-encode just the .php files, setting the property key but without the include-if restriction. Merge those with the already encoded project, and to speed things up, only copy the files being encoded. This replaces the files ending in .php with ones that define the property key but that do not require it to be set by any including file.

```
ioncube_encoder app --into encoded-projects --merge --only-include-encoded-files
        --property key=12345
```

3.11.3 Preventing append and prepend 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, so 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.

3.12 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.

3.12.1 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.

3.12.2 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
```

3.12.3 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
```

3.13 Miscellany

3.13.1 Encoding and Run-time Optimisation [--optimise, --optimize]

By default, the Encoder uses an encoding format that encodes with the best Encoder performance and good run-time performance. Better run-time performance of encoded files may be obtained by increasing the optimisation levels.

The options:

```
--optimise more

--optimise max
```

increase optimisation to either an intermediate or a maximum level.

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

3.13.2 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`

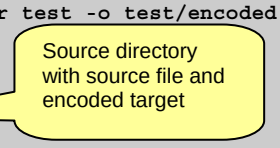
Encoding with the default safety check

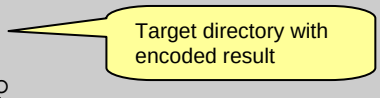
```
$ ioncube_encoder test -o test/encoded

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
```





3.13.3 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.

3.13.4 Program Version [-V, --version]

To display the program version, use:

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

3.13.5 Verbose Mode [-v, --verbose]

Verbose mode will produce details of Encoder operations and progress.

3.13.6 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.

3.13.7 Help [--help]

This option displays a summary of Encoder options.

4 LICENSE GENERATION

Pro and Cerberus versions of the Encoder can restrict scripts to only run in the presence of a license file. Properties can be set in the license file which must match properties set in the encoded files, and also license files can be used to restrict encoded files to a certain machine. A benefit of using license files is that projects no longer need to be re-encoded in order to create products customised for a particular end-user, as all restrictions contained in a license file overrule restrictions set when a file was encoded.

4.1 Creating License Files

4.1.1 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 be unique for a particular product. The output path is the path to which the new license file will be saved.

4.1.2 Using Passphrases to Differentiate Products

As mentioned above, it is recommended that a unique passphrase be used for each product 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 can not unlock files encoded with a second Encoder installation.

4.1.3 License File Format

A license file consists of a header in plain text, followed by an encrypted data block. Properties added to the license file can be optionally exposed in the header block, and can be accessed from encoded scripts using the Loader API.

4.1.4 Server Restrictions

A central feature of license creation is the ability to restrict licenses to particular servers. Encoded files tied to such a license will only run on the permission server. If the encoded files have themselves been created with server restrictions, then the license restrictions take precedence.

4.1.5 Server Restrictions and Expiry Options

Command line options to the license generator which control license expiry and server restrictions are similar to the corresponding options to the Encoder. See section 3.5.1 for details on the syntax. The following options are supported: `--allowed-server`, `--expire-in`, and `--expire-on`. The expiry date can be exposed in the header block by using the `--expose-expiry` option.

It is important to note that restrictions specified when encoding a file are replaced by restrictions specified in a license file, in the case where an encoded file requires a license file. For example, if an encoded file has an expiry time of 2 days, but its associated license has an expiry of 10 days, then the encoded file will stop functioning after 10 days. Similarly, if an encoded file is restricted to a server, but its license has no server restriction, then the encoded file will work on any server, if the license is present.

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

Custom data can be stored in a license file as properties, just as 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 3.11 for more details on the supported syntax. Properties can be exposed in plain text in the license header block by using the option

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

4.1.7 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, then 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 4.2.1 for details on how to customise this behaviour.

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

The text which occurs before the encrypted license data is called the *header block*. It is important that this text is not edited after the license has been generated as 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 an arbitrary line to the header block use the command line option

```
--header-line value
```

4.1.9 Server Data Files [`--use-server-file`]

As with encoded files, licenses can contain server restrictions. To create a license with such a restriction it is necessary to know ahead of time the IP address, MAC address, or domain name of the target server.

To help with this process, one can use the `--ioncube-server-data` Loader API function to generate a *server data file* containing the necessary server parameters. See section 7.2.3 for more details on this Loader API function.

To use a server data file with the `make_license` program specify the command line option

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

where *path* refers to the location of the server data file. It is also necessary to use either the option `--select-server-adapter` or `select-adapters.` See below for more details on these options.

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

If a server data file is created on a server with multiple interfaces, information related to all installed interfaces will be written to the server data file. If PHP reports a domain name and current server IP address then these will also be added to the server data file. The contents of the file can be viewed with the `make_licence` program option

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

The domain name and server IP address are output first, followed by the name, IP address, and MAC address of each adapter installed on the server.

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

A server will typically report an IP address and domain name. If this is the case, then usually the license should be restricted to the corresponding adapter and no other adapter. If the server does not report an IP address and domain name then this may mean that the end user has generated the server data block on the wrong machine (their desktop machine, for example). For this reason, it is recommended to use the

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

option when generating licenses with server data files. This option will use the adapter corresponding to the reported sever IP address, and no other. If no IP address or domain name is reported then the `make_license` program will exit with code 2.

In the general case the option

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

can be used to create a license for an arbitrary set of adapters contained in a server data file. Here *adapter list* is either a comma separated list of numbers which refer to the position of the adapter in the server data file, or the * character. In the latter case all adapters in the server data file will be licensed.

4.1.12 Troubleshooting License Problems

If an encoded script that requires a license fails, the particular error message that is 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 has 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.

4.2 Encoding Files with a License Restriction

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

Use the command line option

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

to specify the path of a license file that should be used to restrict the execution of the encoded script. The Loader will search for the license relative to the location of the encoded script, so a relative path should be used when specifying the license.

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.

4.2.2 Specifying a Passphrase [`--passphrase`]

License files are encrypted using industry proven algorithms. Use the command line option

```
--passphrase pass
```

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. If the Loader cannot decrypt the license file it will prevent execution of the script.

4.2.3 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 4.1.7). 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 section 7. 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.

5 ERROR REPORTING

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

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

This offers easy 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.

6 TROUBLE SHOOTING

6.1 Unable to Start the Encoder

6.1.1 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. When entering the name of a program without a full path, even if your current directory contains the `ioncube_encoder` program it will only be found if '`.`' (dot) is one of the paths listed in the `PATH` environment variable. Dot is an alias for whatever the current directory is.

You can either prefix the path to the program, for example

Run from the current directory

```
./ioncube_encoder
```

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

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

or add the program path to the `PATH` variable.

For example, if using `bash` as your shell, edit the `.bashrc` or `.profile` file that should exist in your home directory and add the directory path to where the program is installed to the `PATH` variable.

Start another shell or type

```
. ~/.bashrc
```

to re-read the `.bashrc` file and for changes to take effect. Read the `.profile` file if you edited that file instead.

6.1.2 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.

7 LOADER API

The ionCube Loader contains an API that provides numerous functions and constants that may be useful to encoded software. The Loader API consists of a set of PHP functions that are available to both encoded and unencoded PHP scripts whenever the Loader is installed. The behaviour of the functions depends on whether the script is encoded or unencoded.

7.1 File Information and Execution

7.1.1 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.

7.1.2 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.

7.1.3 Retrieving Properties Stored in an Encoded File [ioncube_file_properties]

This function returns an associative array consisting of file properties. These items were added to the encoded file with the `--property` command line option to the Encoder.

7.1.4 Retrieving the Loader Version [ioncube_loader_version]

This function returns a string with the Loader version.

7.2 License and Server Information

7.2.1 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 which 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 file is not encoded or has no license file.

7.2.2 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.

7.2.3 Creating a Server Data Block [ioncube_server_data]

When generating a license for an end user it may 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 server.

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

7.3 License Validation

7.3.1 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.

7.3.2 Validating Licensed Servers [ioncube_license_matches_server]

This function returns FALSE if the file is encoded, has 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_matches_server` API function.

7.3.3 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.