



Version 3
for Windows, MacOS and Linux
User Manual

Introduction

An introduction to RubyEncoder

by RubyEncoder Team

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

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

Have fun using your new product...

RubyEncoder 3 for Windows, MacOS and Linux

Copyright © 2023 SourceGuardian Ltd.

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

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

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

Table of Contents

Foreword	0
Part I Introduction	2
1 About RubyEncoder™	2
2 How to buy	2
3 Features	2
Part II GUI manual	7
1 Overview	7
2 Registration	7
3 Welcome screen	9
4 How to start?	9
5 Project	10
Choosing files and destination	10
Locking options	13
Generating a license file	20
Advanced options	22
Encoding	26
6 Installing loaders	28
7 Getting file information	29
8 Preferences and default settings	30
9 Viewing registration information	33
10 Getting help	33
11 Checking for update	35
12 Encoding Ruby on Rails	36
Part III Command line encoder	43
1 Command line tools installation	43
Windows	43
MacOS	43
Linux	44
FreeBSD	45
Docker	46
First run	48
2 Running the command line encoder	48
Note for GUI users	49
Usage	49
Output directory for encoded scripts	51
Specifying which files to encode and which to copy_2	51
Excluding files from processing	52
Encoding entire directory contents_2	52
Locking options (full version only)	53
Advanced options	59

Custom predefined constants	62
Custom error handling	62
Other options	64
Encoding to standard output	68
Exit codes	68
3 Script license generator (full version)	69
Usage	70
4 File information tool (full version)	72
Usage	72
Part IV Running protected scripts	75
1 Installing loaders	75
2 rgloader directory structure	75
Part V Common mistakes	78
1 Encoded scripts modification	78
2 Error messages during encoding	78
3 Error messages when running protected scripts	79
Part VI Advanced users	85
1 Ruby shell scripts encoding	85
2 Marking a file to be skipped during encoding	85
Part VII License	87
1 RubyEncoder License	87
2 RubyEncoder Loaders License	92
Part VIII Changelog	96
1 Version 3 / February 2023	96
2 Version 2.7.5 / February 2022	96
3 Version 2.7 / May 2020	96
4 Version 2.6 / November 2019	97
5 Version 2.5 / January 2019	98
6 Version 2.4 / February 2017	103
7 Version 2.3 / December 2016	104
8 Version 2.2 / February 2015	105
9 Version 2.0 / September 2013	106
10 Version 1.5 / March 2013	107
Index	110

RubyEncoder 3 **for Windows, MacOS and Linux**

Part



1 Introduction

1.1 About RubyEncoder™



RubyEncoder has been built as professional system for source code protection. Our team of programmers, some of whom programmed and developed the successful SourceGuardian for PHP Encoder (www.sourceguardian.com) have created proprietary methods for encrypting code whilst keeping the maximum flexibility for the distribution of your scripts.

We, as a team, are very excited about the potential for our product as well as the Ruby and Ruby on Rails community. We finally have a method to protect and distribute our commercial code and many of the developers who have worked with us during the beta phase have said that RubyEncoder solves many of the problems they previously had.

As for the future of RubyEncoder, we thank everyone who has purchased, downloaded or even taken the time to browse our site. We plan to continue to increase the functionality and power of these programs whilst keeping an affordable upgrade path. We always welcome new suggestions and if you feel that you have something you would like us to add, then feel free to contact us

Thanks for your interest, and thanks for your business.

The RubyEncoder Team

1.2 How to buy

To purchase RubyEncoder 3, please visit the following:

Website: http://www.rubyencoder.com/buy_now.html

There are two methods available: via Credit/Debit Card or via Paypal.

1.3 Features

Protection method

The RubyEncoder 3 protects Ruby scripts by compiling Ruby source code into a bytecode format and this is followed by encryption. This protects your scripts from reverse engineering. Ruby scripts protected by RubyEncoder can be executed but cannot be read and do not contain original source code within encoded files in any form. Scripts protected with RubyEncoder require installation of a RubyEncoder Loader in order to run. The RubyEncoder Loader is a compiled Ruby module which is automatically loaded and used to run protected scripts. RubyEncoder Loaders are binary files that differ for OS and platforms. See the [protected scripts loaders](#) section below to know more about RubyEncoder Loaders.

To protect your scripts from unauthorised usage RubyEncoder 3 has added features that can optionally lock your scripts to run only from predefined IP addresses, domain names, LAN hardware addresses (MAC), machine ID. RubyEncoder 3 can also easily produce trial versions of your scripts by setting an

expiry date for the script or by limiting the number of days that protected script will work. With RubyEncoder 3 you may optionally lock your scripts so that they require a special license file in order to run. This file may be distributed with the script or separately from it and this option gives you an opportunity to encode your scripts once and distribute to users with different licenses. Each license may have different and specific attributes. The RubyEncoder license generator tool may be used to create a license file directly on your server. This lets you easily provide your customers with a license file for your product in the moment of purchase or downloading of a trial version of your product.

Protected code

This sample Ruby code:

```
puts "Hello World!"
```

Becomes like this after encoding by RubyEncoder 3:

```
# RubyEncoder v2.7
if !defined?(RGLoader) || !RGLoader.respond_to?(:check_version) || !RGLoader.
check_version('2.7') then _d=_d0=File.expand_path(File.dirname(__FILE__)); while true
do _f=_d+'/rgloader/loader.rb'; load _f and break if File.exist?(_f); _d1=File.
dirname(_d); if _d1==_d then break if defined?(RGLoader); raise LoadError, "Ruby
script '"+__FILE__+"' is protected by RubyEncoder and requires a RubyEncoder loader
to be installed. Please visit the https://www.rubyencoder.com/loaders/ RubyEncoder
web site to download the required loader and unpack it into '"+_d0+"/rgloader/'
directory in order to run this protected file."; exit; else _d=_d1; end; end; end;
RGLoader_load
( 'AAUAAAAEeAAAAIAAAAA/9CRIInRdYGF08QzqDs6IsFobMMmTvASfFmPSuXukUFstIrnX2/
lLKzvHNGlw6DKHDPEg9Zx5BUX09tKczF3aEa0+GPEMkU4LtUb26+crViJfaMMwUZ3AC2vpA+
+8bqUgorzhV8hMA14u6tYk6P+lhZ6SvZEt8ZxsAAAAFAAAAAaAAACoZ/kji4TUphCWdUg13zy4/RdrY8pCK/
lnvealjd9F1FSOXRq2gLO/BlnAAkp8Pg4XrEcVeSgQGFiIhjbM4EJkXILfiY9ebia/
M2rClLBxGpmLSP8AXWMBaK112Qo+jkVafNZ//R3ZqAAAAA== ' );
```

Supported Ruby versions

RubyEncoder 3 supports encoding for Ruby 1.8.7, 1.9.x, 2.x. The encoder can encode your Ruby files for multiple versions of Ruby at the same time. When encoding for multiple Ruby versions your Ruby scripts must be compatible with the specified versions of Ruby language. RubyEncoder writes separate bytecodes into the protected script for each version of Ruby. The RubyEncoder Loader will extract and run the required version of bytecode from the protected script during the protected script execution.

Interface

Versions for MacOS, Linux and Windows include a graphical user interface for the encoder and tools. The command line tools are also included.

Locking

To protect your scripts from unauthorised usage RubyEncoder 3 has added some features that can optionally lock your scripts to run only from predefined IP addresses, domain names, LAN hardware

addresses (MAC) or machine ID. RubyEncoder 3 can also easily produce trial versions of your scripts by setting an expiry date for the script or by limiting the number of days that the protected script will work. To protect against local date change for trial version of your protected scripts there is an option for time checking with atomic online time servers. With RubyEncoder 3 you may optionally lock your scripts so that they require a special license file in order to run. This file may be distributed with the script or separately from it and this option gives you an opportunity to encode your scripts once and distribute to users with different licenses. Each license may have different and specific attributes. The RubyEncoder license generator tool may be used to create a license file directly on your server. This lets you easily provide your customers with a license file for your product at the precise moment of purchase, or downloading of a trial version, of your product.

Here is a sample list of features:

- locking to date with optional atomic online time servers checking
- locking to multiple domain names
- locking to multiple IP addresses
- locking to multiple LAN hardware (MAC) addresses, also works for CLI scripts
- locking to machine ID, also works for CLI scripts
- protecting of CLI scripts with remote verification URL
- improved locking to a specific domain name with encryption. The domain name is used as a part of the key for encryption, so protected scripts may not be decrypted and run from another domain.
- improved locking to the ip address with encryption. The ip address is used as a part of the key for encryption. This means that protected scripts cannot be decrypted and run from another ip address.
- improved locking to the machine ID with encryption. The machine ID is used as a part of the key for encryption. This means that protected scripts cannot be decrypted and run from another machine.
- locking to an external license file produced by the built-in RubyEncoder 3 license generator. This is a deal for creating protected scripts to be deployed to different users and it even allows to assign different locking options to different users. The RubyEncoder 3 license generator tool can run from GUI or as a command line tool which adds another powerful element - It provides a method for licenses to be dynamically generated and this would be useful (for example) when selling scripts online.
- locking protected scripts to work only online

Other options

The following is not an exhaustive list, but covers some of the other options in version 3:

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

be included here

- Please see the [change log](#) for further details

Cross platform

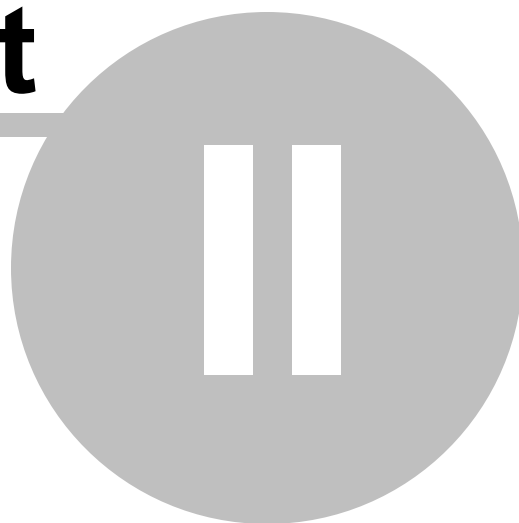
Cross platform encoding. A script encoded under one operating system will run under any other supported operating systems. Currently we have a GUI and CLI encoder for Windows, MacOS, Linux, CLI encoder for FreeBSD Script Loaders will run under MacOS, Linux, FreeBSD, MinGW and Windows. Also we have plans to support more operating systems for protected scripts execution.

Evaluation

We provide a Free 14 days evaluation of RubyEncoder 3 Locking options are not available in this free version.

RubyEncoder 3 **for Windows, MacOS and Linux**

Part



2 GUI manual

2.1 Overview

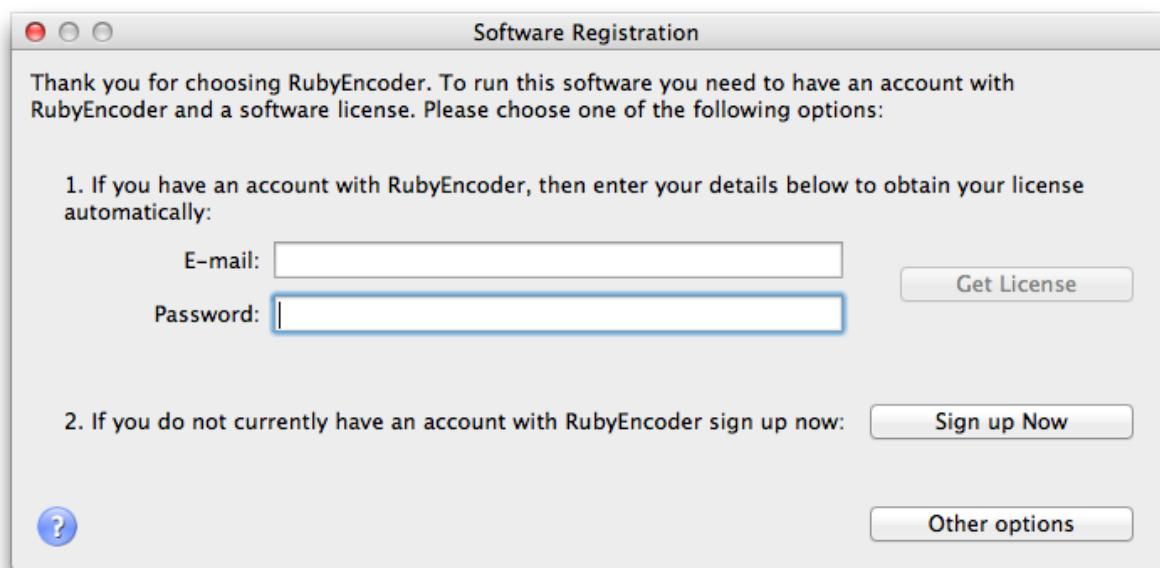


The RubyEncoder™ 3 is a GUI application which runs on Windows, Macintosh or Linux computers depending on the version of RubyEncoder you are running, user-friendly graphic interface to our command line encoder. It uses all the powerful features that command line encoder offers, while adding many additional useful enhancements to the encoding routine. User interface is easy to use and allows access to all powerful features of the encoder. The GUI has a multi-document interface which lets you work with multiple RubyEncoder projects at the same time.

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

2.2 Registration

On your first run of RubyEncoder™ 3 you should see following screen:



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

1. Obtaining a license from the application itself.

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

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

When you purchase a full version of RubyEncoder™ 3, or request a demo version of it, you will receive an email with details on how to access your profile on our web site. This email contains a user name (which is your email address entered during registration) and a profile password. Please type them in the 'Email' and 'Password' fields and click on 'Get License' button. After the license has been successfully downloaded RubyEncoder™ 3 will run and you see a welcome screen. If anything has gone wrong with the installation, you will see an error message again. Make sure you have entered your email and password correctly, check your Internet settings and try again. If you cannot get a license please try another option ([see 2](#)).

2. Obtaining a license via the 'User Account' section on our web site.

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

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

3. If you want to obtain license manually (in the case this program unable to contact our site) please enter Registration Code shown below to your profile. Then download the license file and save it to your local disk.

Registration Code:

4. If you have a license file on your local disk, then click to select it:

Once you have entered your user profile on the web site, select and copy the registration code from the RubyEncoder™ 3 application (by clicking on 'Copy'). Paste it to a corresponding field in the 'Available licenses' section in the user account area on our web site.

When you have done the above, click on 'Create License' in the user account in order to create and download a license file. If the download does not start automatically, click on 'Download'. Save the license file to your local disk. After that, click on 'Browse' button in the RubyEncoder™ 3 application, select the downloaded license file. Software Registration window will be closed after successful license registration and a welcome screen appears.

2.3 Welcome screen



This welcome screen appears when you start RubyEncoder™ 3 and there are no project files opened yet. If there are projects opened, you may open Welcome screen in Window/Open Welcome Screen menu. The following outlines features available on this screen:

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

Click on "Open Existing Project" if you already have a RubyEncoder™ 3 project and want to continue working with it, run encoding or generate a script license.

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

2.4 How to start?

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

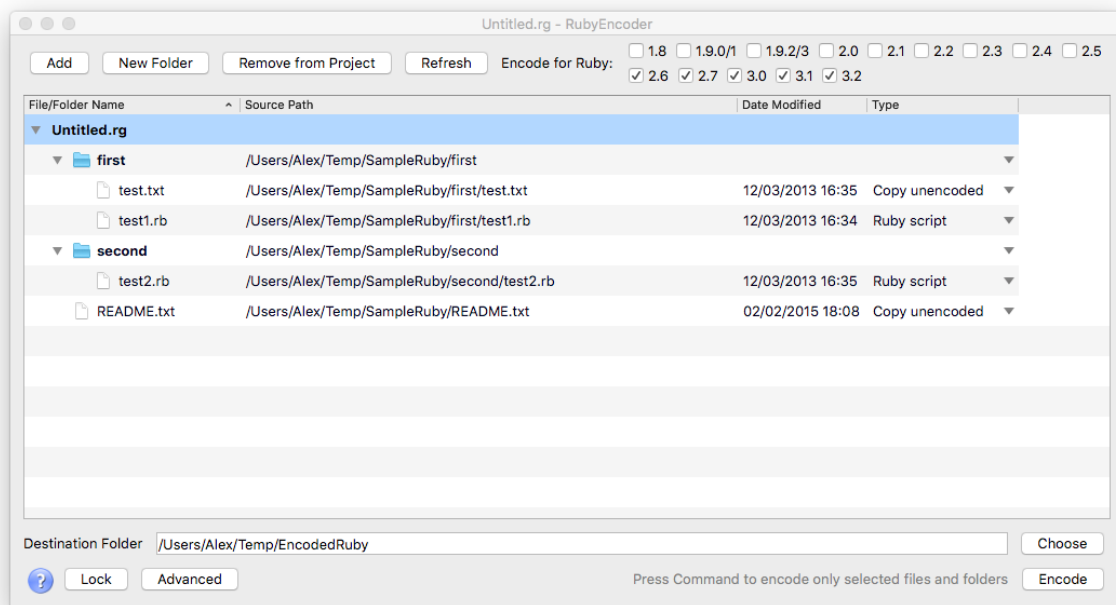
- 1) create a new RubyEncoder project
- 2) add files to the project tree
- 3) select versions of Ruby your code is compatible with
- 4) select a destination folder
- 5) click Encode

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

If you want to start encoding of your Ruby on Rails project, you may jump directly to [Encoding Ruby on Rails](#) topic.

2.5 Project

2.5.1 Choosing files and destination



A project window is the main window you will use when working with RubyEncoder encoder. It displays a RubyEncoder project which includes information about files to be encoded, encoding mode, target Ruby versions, locking and advanced options. You can save the project to a file using File/Save or File/Save As menu item. You may load a previously saved project using File/Open or choose from your recent projects using File/Open Recent menu item.

You need to create a RubyEncoder project before you can encode files. This is a simple process during which you will choose files to be encoded, select the destination folder, choose target Ruby versions, encoding mode and set encoding options. First two steps are required. All the other steps are optional.

1. Choose files for encoding.

Clicking on "Add" displays a standard dialog for selecting files or folders. You may either add separate files or add entire folders. You may set encoding mode for files or delete files from the project that you do not want to encode. No real deletion happens when you delete files or folders from the project using "Remove from Project" button. Clicking on "Remove from Project" simply excludes a file or folder from encoding.

Choose encoding mode for files or folders. There are 3 modes available:

- Ruby Script - the file will be encoded as Ruby script
- Skip - the file will not be encoded or copied to the destination folder. It's useful for files or folders that you do not want to encode, but want to keep in the project tree.
- Copy unencoded - the file will not be encoded. It will be copied as is to the destination folder. It's useful for files that you want to include to the protected project but which you do not want to encode at all, e.g. text, javascript, configuration, images, audio, video etc files.

To change encoding mode for a file or folder you need to choose it from the Type dropdown in the files list. Double click to select another encoding mode. You may select the encoding mode for each file separately or set it once for a folder. When you select the encoding mode for a folder, it will be assigned to all files and subfolders recursively.

When you add new files to the project, encoding modes are set automatically. This is done according to default settings in [Preferences](#). You may change defaults there if you often need to make changes in encoding modes after adding new files.

You may build a new "tree" of your project instead of adding existing files and folders. Use "Create Folder" button to create a new folder within the project. You may change the Project Window Mode in [Preferences](#). You may add files and folders to the newly created folders as usual. Files added to the project will be copied to the destination folder and encoded there replicating the folders structure of the project.

2. Choose a destination folder.

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

3. Optionally set target Ruby versions.

RubyEncoder encoder produces different bytecodes for different Ruby versions to provide maximum compatibility and full support of Ruby language features. So you need to choose target Ruby versions for encoding. All versions of the internal bytecode for one source script will be packed into one protected script. Choose target Ruby versions by clicking checkboxes on the top of the project window. A new project will use default settings. You may change the default setting in [Preferences](#).

Your choice should be easy when you know what version of Ruby is installed on the server where you plan to run your scripts. It is important to know which Ruby versions your code is compatible with and do a correct choice. Nothing bad will happen if you do a wrong choice. A built-in Ruby compiler will try to compile your code for each of the selected versions of Ruby and display an error message in the case your code is not compatible with any of the selected versions. Note, the entire script will NOT be encoded in that case and you can find details in the encoding log.

4. Optionally set locking options.

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

5. Optionally set advanced options.

You may set advanced options which include character encoding for Ruby 1.9.x and 2.x, custom header code, loader not found error handler etc. Click Advanced to set advanced options. Please refer to [Advanced options](#) section for further details.

6. Click Encode to start encoding.

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

Files and folders highlighting in the project tree

Files and folders may be shown highlighted in the project tree. The legend is:

Folders - bold, virtual folders - green, files/folders changed or added since last encoding - blue.

2.5.2 Locking options

Lock Options

☒ Lock to a license file
 Enter license file name:

☐ Set expiration date
☒ Script will expire in (days)
☐ Script will only work online
☐ Use remote verification (for CLI scripts)

Run from these IP addresses only

IP addresses	Mask
192.168.000.100	255.255.255.000

☐ Encrypt with IP (single IP only)

Run from these domains only

Domain Name

☐ Encrypt with domain (single domain only)

Run from these Machines only

Machine ID

☐ Encrypt with machine ID (single machine ID only)

Online time servers for date check

Time or NTP server
pool.ntp.org

☐ Check date with online time servers

Custom defined constants

Name	Value
Copyright	MyCompany

Run from these MAC addresses only

MAC address

Custom error handlers

Error Condition	Function Name
All errors	▼ my_error_handler

This window is used for setting locking options for encoded scripts. All the locking options are stored internally encoded within your protected script. You may choose "Lock to external license file" option to put all locking settings into a special encoded file which acts as a license file for running your protected scripts. You need to specify a [license file name](#) in order to use this option.

[Set expiration date](#)

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

Script will expire in (days)

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

Online time servers for date check

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

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

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

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

Run from these IP addresses only

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

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

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

As a developer you may wish check if SERVER_ADDR or LOCAL_ADDR variable is available in the

environment on the target server by printing ENV['SERVER_ADDR'] or ENV['LOCAL_ADDR'] values

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

Encrypt with IP

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

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

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

Run from these domains only

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

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

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

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

Example 1: mydomain.com

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

Example 2: www.mydomain.com

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

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

Example 3: *.mydomain.com

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

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

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

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

Encrypt with domain

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

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

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

Run from these MAC addresses only

Lock the script to LAN hardware (MAC) addresses (The "MAC" word used here has nothing about Apple Macintosh computers. MAC address is a hardware address of the local area networking LAN controller available for all platforms). This address is usually unique for every networking adapter and so it may be easily used to identify a machine. A MAC address is 6 bytes long, with each byte represented in hex and separated with a colon (:). The encoder will lock a script to run only from the machine which has a networking adapter with a specified MAC address. If there is more than one LAN adapter installed then script will check all of them. If an attempt is made to run the script on a machine that is missed a correct adapter, then the script fails with the error message: "This script is not licensed to run on this machine". You may specify multiple MAC addresses, if any one address matches then the script runs.

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

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

Run from these machines only

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

Use `RGLoader::get_machine_id()` API method and get the machine ID of the computer where you run this method. It may be called directly or from the encoded Ruby code (not locked with any options). Note, as this API method is binary compiled into the loader, an appropriate RubyEncoder loader must be installed and loaded (require directive) before your code calls this method. We recommend that you create a mini project, encode it and include the loaders for obtaining the machine ID from the client's machine before locking to it, in that case loaders will be found automatically.

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

Encrypt to machine ID

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

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

Remote verification

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

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

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

```
print RGLoader::get_verification_id()
```

Note 1: if you use any frameworks, you may need to use another mechanism for sending a string to the output stream instead of `print`, e.g. `res.write` for Cuba etc

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

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

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

Example: http://mydomain.com/verification_id.rb

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

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

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

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

Script will only work online

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

Custom defined constants

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

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

To get a predefined constant value from the **encoded script** use `RGLoader::get_const()` RubyEncoder API function. This function is defined in the RubyEncoder loader and returns a predefined RubyEncoder constant value or nil if constant with the specified name is not defined. RubyEncoder constants names are **case sensitive**.

Syntax: `string RGLoader::get_const(string)`

Additionally there are 5 constants predefined for all protected scripts:

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

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

Custom error handlers

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

```
error_handler( code , message )
```

You may use any name for this function. Also you may have different functions for different script errors. The first argument will contain an error code. The second one will contain a default error message. To set a custom error handler use --catch option for the [command line rubyencoder command](#) or set your error handler on the [Locking screen](#).

There are following error handler conditions defined:

Err tag	Returned Default message Code
Restricted to run	01,02,03, This script is not licensed to run on this machine. 04,05 (code indicates a reason: 01-IP, 02-domain, 03-MAC address, 04-machine ID, 05-remote verification URL locking)
License broken	06 A license file which is required to run this protected script is invalid...
License expired	09 This script has expired...
License not found	13 This protected script requires ... license file in order to run
Running offline	20 This protected script requires the Internet connection in order to run
All errors	- -

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

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

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

External license file name

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

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

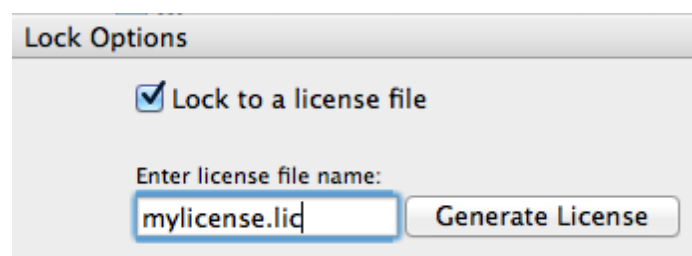
Locking your scripts to an external license file gives additional protection to your scripts. Using locking to a license file is the best if you need to deploy one script or the entire project to different users, but need to use different restriction options for each of them. Find more about how it works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

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

Generating a license file

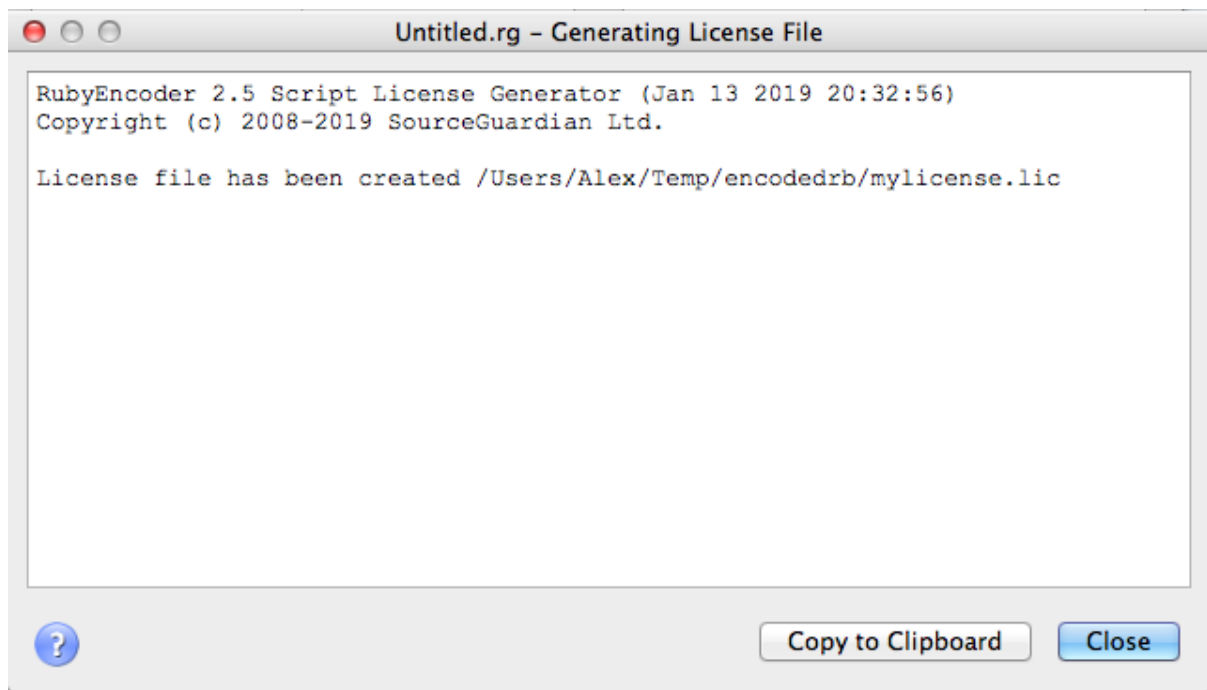
To generate a license file click on "Generate License" button. Please refer to [Generating a license file](#) section for further details.

2.5.2.1 Generating a license file



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

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



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

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

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

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

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

2.5.3 Advanced options

Advanced Options

Project ID

Project Key

Encoding

☒ Stop on compiler errors

☒ Rails compatibility mode

☐ Do not integrate dynamic loader

Custom 'loader not installed' error code

Custom header code, error handler or copyright

License file custom text

☐ Only encode files changed since last encoding

The project was last encoded on: 06.02.2017 12:21:55

Additional command line options

Project ID

This field lets you assign ID to your project that is used to identify what license it should accept. Specify the same Project ID in the license generator when you generate a license file for this project (if you use a command line license generator tool). This option is useful when you want to deploy several products that use external license locking so that each license works only with the corresponding Project ID.

Project Key

This field is used in conjunction with Project ID and is required if you use external license locking. RubyEncoder license file locking algorithm uses an idea of two keys. The first key (Project Id) is stored within the encrypted area of protected scripts and used to decrypt an external license file. The second

key (Project Key) is stored within the license file and used to decrypt the bytecode from the protected script. This algorithm protects your product against creating a full working copy from the demo version by some people who may be interested in this. In order to decrypt and run the protected script a valid license file for the full version of your product is required. Otherwise it's impossible to decrypt and run the bytecode. Project Id and Project Key values are required if external license protection method is selected.

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

Encoding

Specify a target character encoding for Ruby 1.9.x and newer. This option does not make any sense for Ruby 1.8. RubyEncoder compiles source Ruby files to a binary representation, as a result 'magic comments' cannot be used for specifying character encoding of source files as they are read on the 'running' stage, not during compilation. Please use this RubyEncoder option when you are encoding files that have 'magic comments' in the code for specifying the character encoding.

Example 1: *UTF-8*

Example 2: *ISO-8859-1*

If not specified, UTF-8 is used by default

Rails compatibility mode

Enables Rails compatibility which lets you encode all Rails *.rb files (you can encode only pure Ruby files with RubyEncoder). Normally you can encode only application controllers, model and helper files. Other files if encoded would not work under Rails if the Rails compatibility mode was not used. If this option is selected, a more relaxed CRC checking will be used for protected scripts. This lets Rails engine use eval() for loading protected files.

Stop on compiler errors

This option instructs the encoder to stop encoding at first critical error. This may be useful if you have many files in the project and there is a risk of missing errors and leaving some files unencoded because of that.

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

Don't integrate dynamic loader

You may use this option if you don't want to include the default starter code into protected scripts. Scripts encoded using this option will not be able to automatically find and load an appropriate RubyEncoder loader and you have to start an appropriate loader before running the encoded script. This option is useful if you have multiple encoded files and want to start the RubyEncoder loader manually from your code maybe from the custom folder before running the protected files. When starting loaders manually you may either 'require' the loader.rb helper file which is available along with binary loaders or

you may load the binary loader directly.

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

Keep modification date for encoded files

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

Custom header code

You may add any code and it will be inserted BEFORE the protected scripts code starts within protected files. This custom header code WILL NOT BE ENCODED. This should be syntactically correct Ruby code. Of course, it may be Ruby comments starting with a # symbol. This option is useful for including copyrights into protected scripts, adding [custom error handlers](#) for RubyEncoder errors.

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

{RG_DATE} - current date i.e. date of encoding

{RG_LICENSEE} - RubyEncoder license owner from the RubyEncoder license file

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

Loader not installed error code

By default a RubyEncoder protected script will generate an exception with the following message when the RubyEncoder Loader is not installed:

Ruby script '...' is protected by RubyEncoder and requires the RubyEncoder loader. Please visit the <http://www.rubyencoder.com/loaders/> RubyEncoder site to download the required loader and unpack it into '.../rgloader/' directory to run this protected script. (RuntimeError)

It is possible for you to change the default loader error behavior. This option allows you to change the default error action of the protected script if it cannot find or load an appropriate RubyEncoder Loader file. You may use any Ruby code here and it will be executed as a replacement to the default RubyEncoder loader exception. This code WILL NOT BE ENCODED. If you want you may load the required RubyEncoder Loader with "require" directive from a non-standard directory instead of printing an error message.

Example:

```
puts "Loader is not found. Call 123-456-7890 for support"; exit(1);
```

The encoded script will have the defined code as unencoded:

```
# RubyEncoder v1.0

_d = _d0 = File.expand_path(File.dirname(__FILE__)); while 1 do _f = _d + '/rgloader/
loader.rb'; bre
ak if File.exist?(_f); _d1 = File.dirname(_d); if _d1 == _d then

puts "Loader is not found. Call 123-456-7890 for support"; exit(1);

break; else _d = _d1; end; end; require _f; RGLoader::load
( 'AAEAAAAEaAAAAIAAAAA/6vBQ2j8ivZCR3gVFB15
Hr/Y90fXBYLdnQDavAjb9qL66uNG0QBaN4Uk9NrQUBHzhv/
WHM97yUQkyjHsiD9nsMr9JiGEavcQ5tAIbHD1/lxoxia2T0rPle4V
e8
+H+3odwWqOIjks94QLEgAAAGAAAAB75w3qzOZQcmvQDuOs+G9ZVhz0GbAyloT4injTT83Ii jyj0iubOUfKRn2
+frb7QOm3dEXG
qgUtKkGzz2yaCE0T9yV1V0ZtZv4RKezGylUJdYtDb4lrK0t8S9FRLBYnwAYAAAAA' );
```

Now a test run without a required RubyEncoder Loader installed:

```
>ruby hello.rb
```

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

License file custom text

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

All user {constants} that are defined with the [custom constant option](#) will be replaced in the text. Also some standard RubyEncoder constants may be used:

{RG_DATE} - current date i.e. date of encoding
 {RG_LICENSEE} - RubyEncoder license owner from the RubyEncoder license file, i.e. your name.
 {RG_EXPIRY_DATE} - expiry date of the custom license you generate. If the expiry date is not set, a word "never" will be placed into the text

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

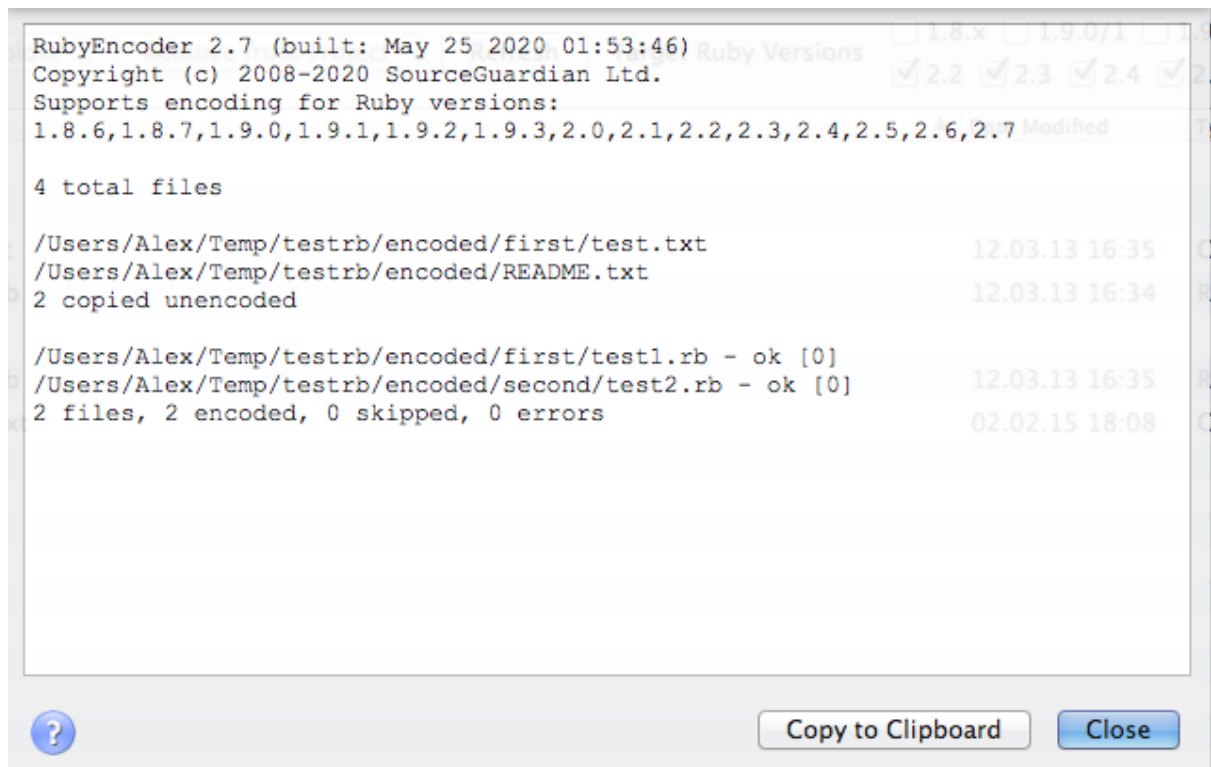
Only encode files changed since last encoding

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

Additional command line options

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

2.5.4 Encoding



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

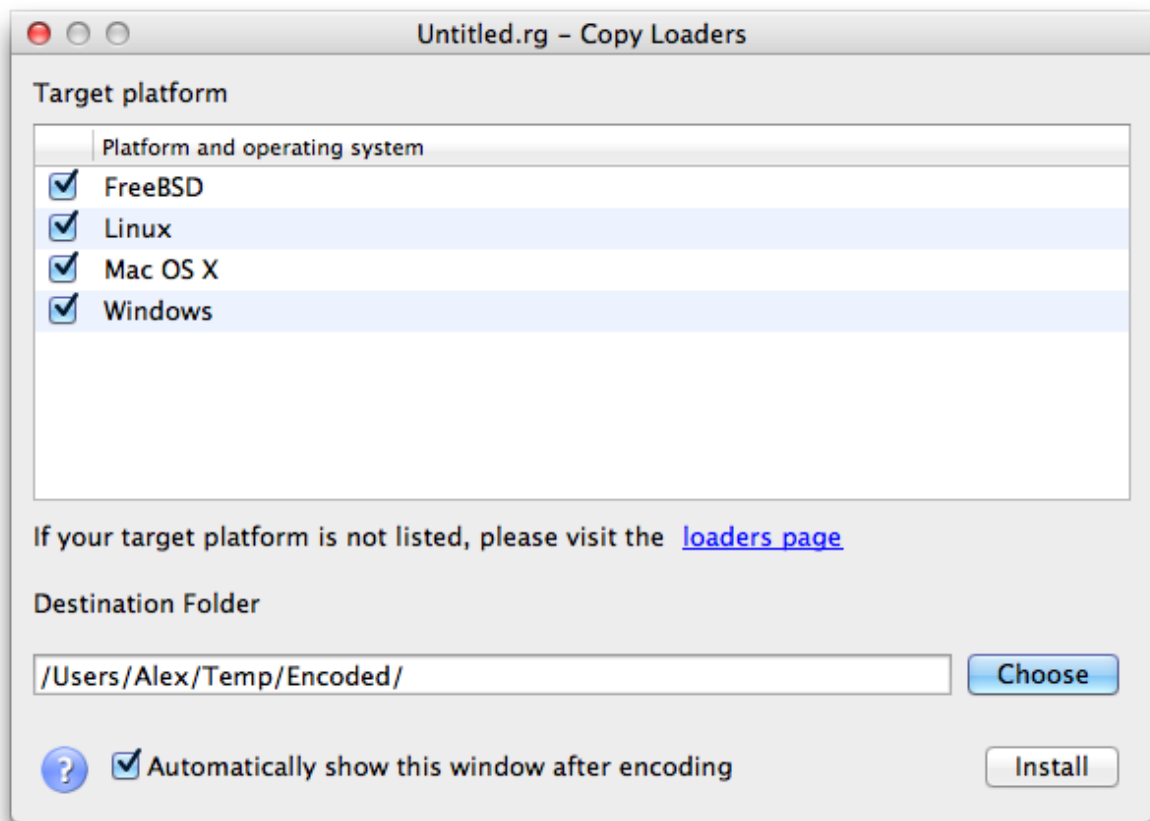
The following error messages may appear in the log next to the file names:

Error message	Description
ok	The script was encoded without problems.
file not found	File for encoding was not found (this error should not appear in the GUI).
Ruby compiler error	You have an error in your script and the encoder could not compile it into bytecode. This message is specific to the Ruby version which is also displayed as well as the line number. Check if your script is free from errors and that it is compatible

	with all the versions of Ruby you are encoding for. E.g. using of some new language features are only possible with recent versions of Ruby. Encoding such scripts for older versions may result in the error.
could not write file	The destination folder that you selected is not writable.
file is already processed by RubyEncoder	File is already encoded with RubyEncoder and could not be encoded once again (this error should not appear in the GUI).
empty file, skipped	Empty files could not be encoded and not processed. If you need to have this empty file in the destination folder, change encoding mode to "Copy unencoded" for it.
not regular file, skipped	The file you have selected for encoding is not a regular file (for example it's a device driver file etc)
copied	This is normal. You have selected "Copy unencoded" for this file and it was copied to the destination folder.
internal encoder error	Please let us know about this error. Send us a support ticket including the file that caused this error.
unknown error	Please let us know about this error. Send us a support ticket including the file that caused this error.

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

2.6 Installing loaders



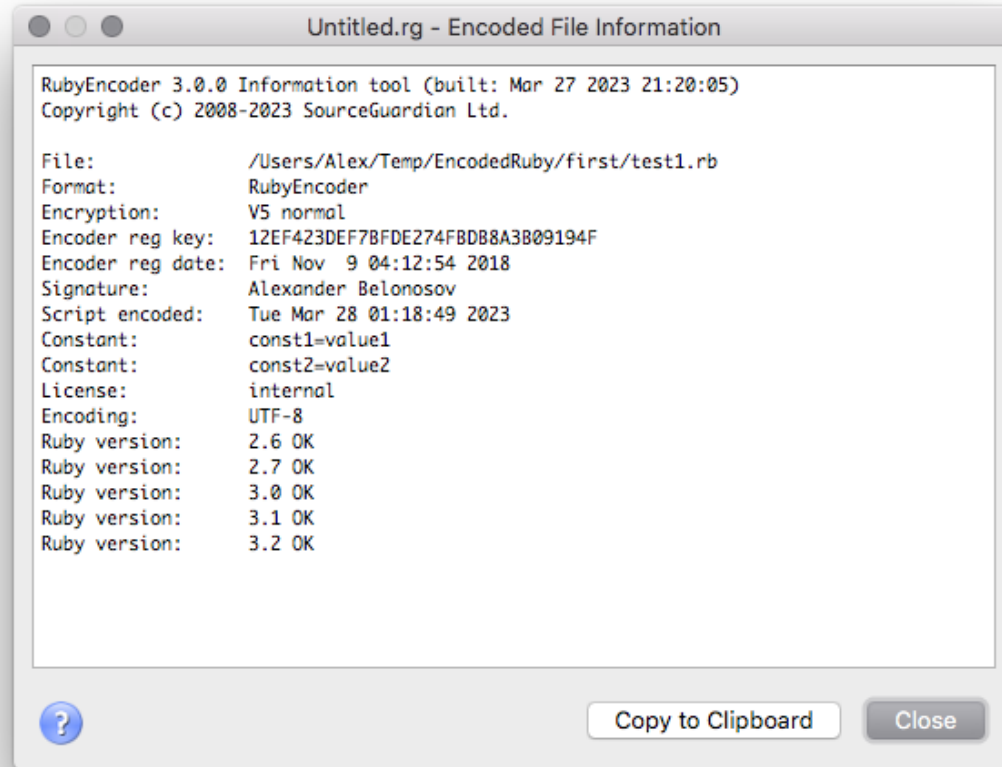
In order to install loaders to the destination encoded scripts folder, please use File/Loaders Installation menu item. Select loaders to install, choose the destination folder and click Install. Loaders will be copied to the /rgloader/ subfolder within the destination folder.

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

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

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

2.7 Getting file information



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

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

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

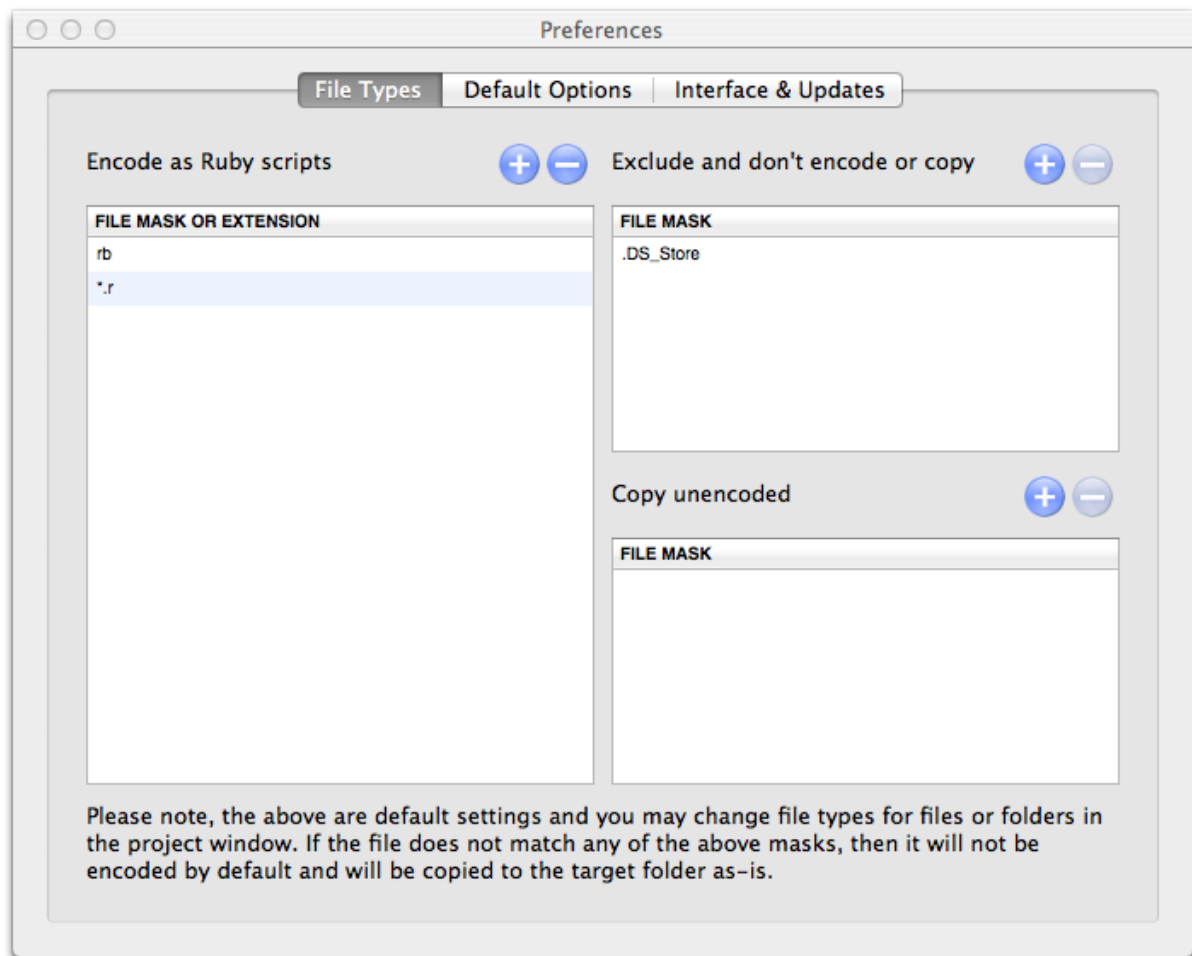
If you prefer to use the command line for getting file information, please use [rginfo command line tool](#).

2.8 Preferences and default settings

Choose Preferences from the File menu to display Preferences window.

Mac OS users - choose Preferences from the application menu or press Command + , (comma)

File Types



File types tab allows you to set file types which will be used for setting default encoding mode when you add new files to the project. Also you can specify file types which will not be added to a project when you add multiple files or folders. Use "Encode as Ruby scripts" list to set file types for encoding using Ruby script mode by default. You may add a file extension without a period to the list or use * or ? wildcards.

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

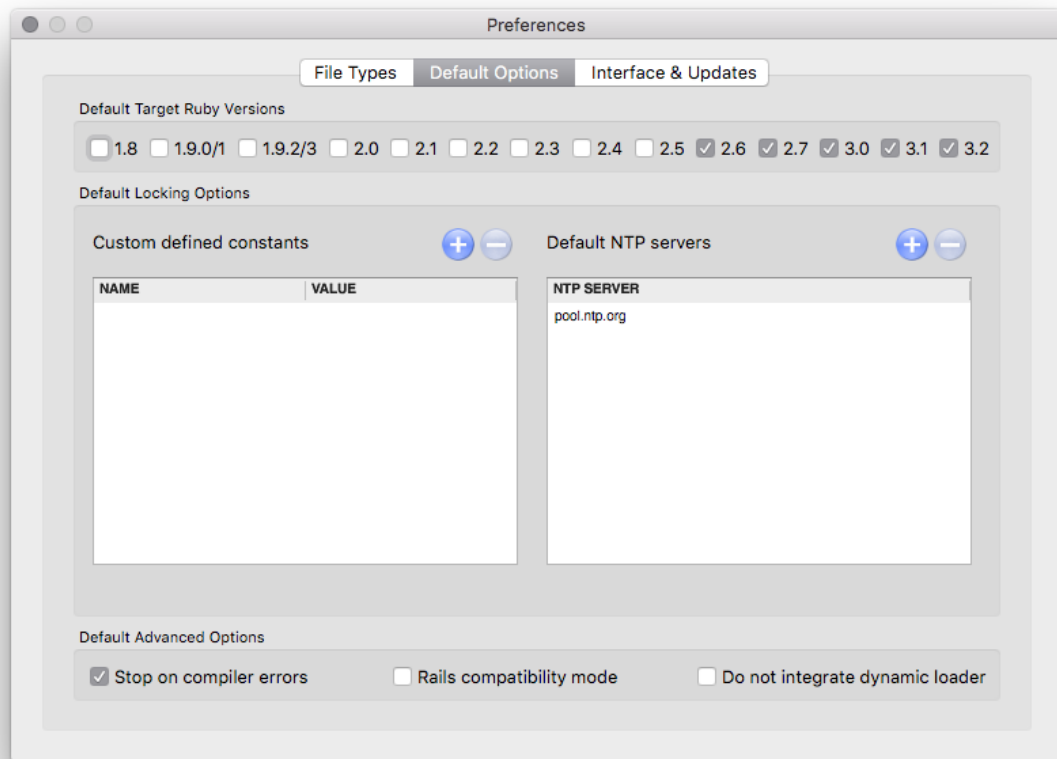
Use the "Copy unencoded" list to configure files which must be copied to the destination folder as-is without encoding. This is useful if you don't need to encode some of the files but still need them to be

copied. They may be templates, configuration files, javascript, texts, images etc.

A default setting is to encode .rb files as Ruby scripts.

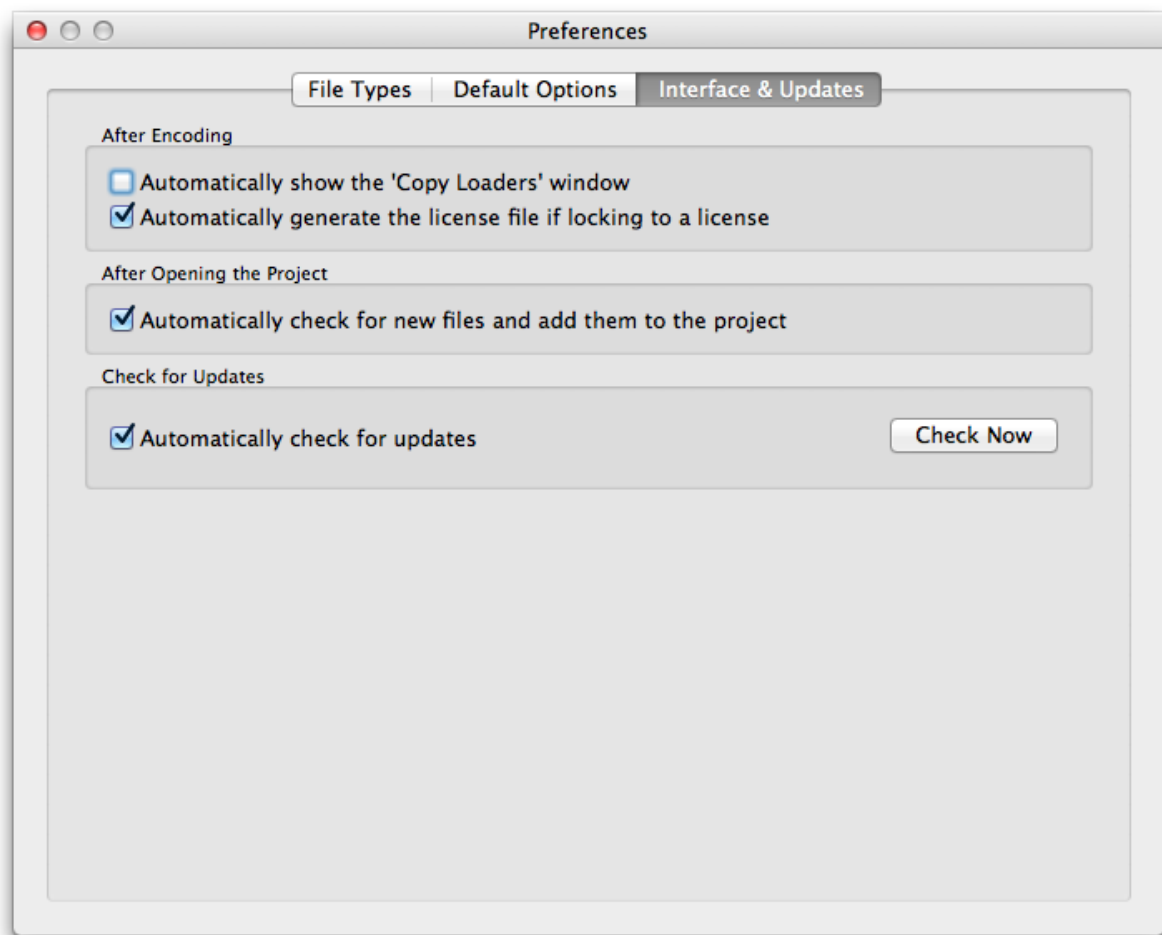
Please note, the above lists only define the default behavior and assigning encoding mode for newly added files. You may always change encoding mode for files and folders directly in the [project window](#).

Default options



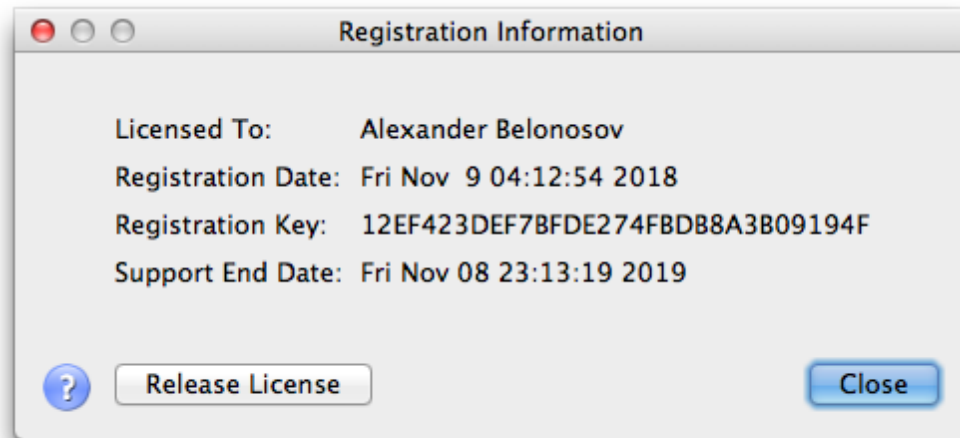
Default options tab allows you to set default encoding options which will be used for new projects. You may set default target Ruby versions, define custom constants, define a list of online time servers you prefer to use for date checking, set default [advanced](#) options. This tab is useful for setting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

Interface & Updates



Use Interface & Updates tab to setup if the "[loaders installation](#)" window will be shown after successful encoding. Use "Check for updates" option to enable or disable automatic checking for new versions of RubyEncoder. Click on "Check Now" to check for the update manually.

2.9 Viewing registration information



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

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

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

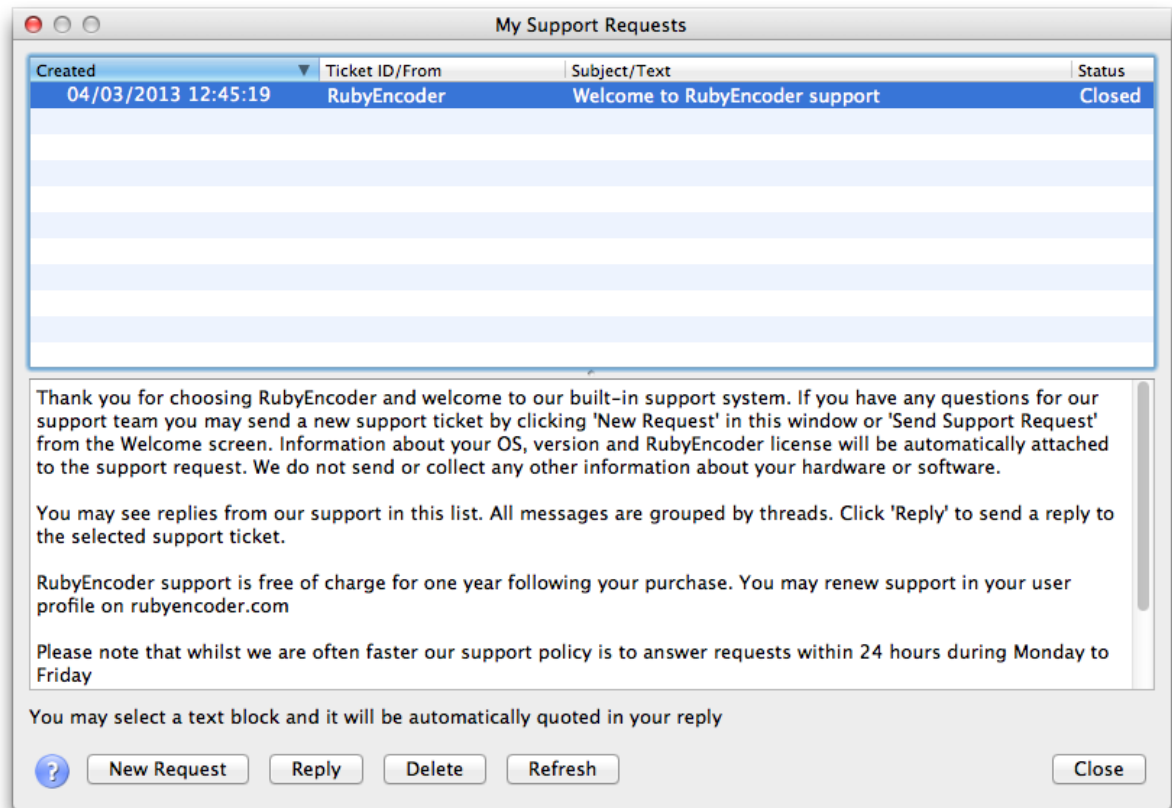
2.10 Getting help

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

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

Built-in support

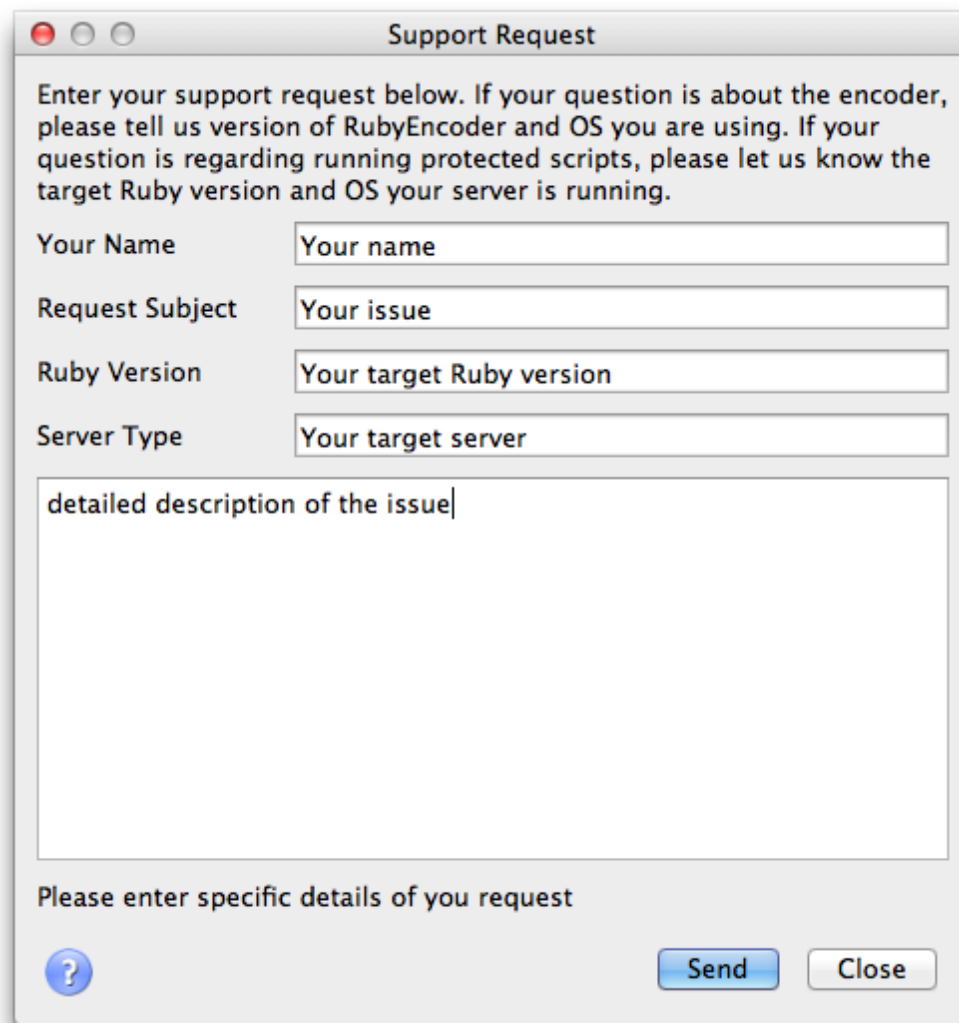
RubyEncoder GUI includes the built-in support window which lets you easily send your questions to our support team, get answers and track issues - all in one window. Select "My Support Requests" from the Help menu to open built-in support window. You may also send a ticket directly without opening the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



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

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

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



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

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

2.11 Checking for update

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

We update loaders when a new official version of Ruby is released. To check for loaders update please

visit the loaders page on our web site by selecting Help/Download Latest Loaders or click to open this link in your browser <http://www.rubyencoder.com/loaders/>

2.12 Encoding Ruby on Rails

It's very easy to encode Ruby on Rails application with RubyEncoder. And to keep this process simple we don't cover any advanced options here. After trying to encode your Rails application and running it, you may review other options of RubyEncoder starting from the [Project](#) section in this manual.

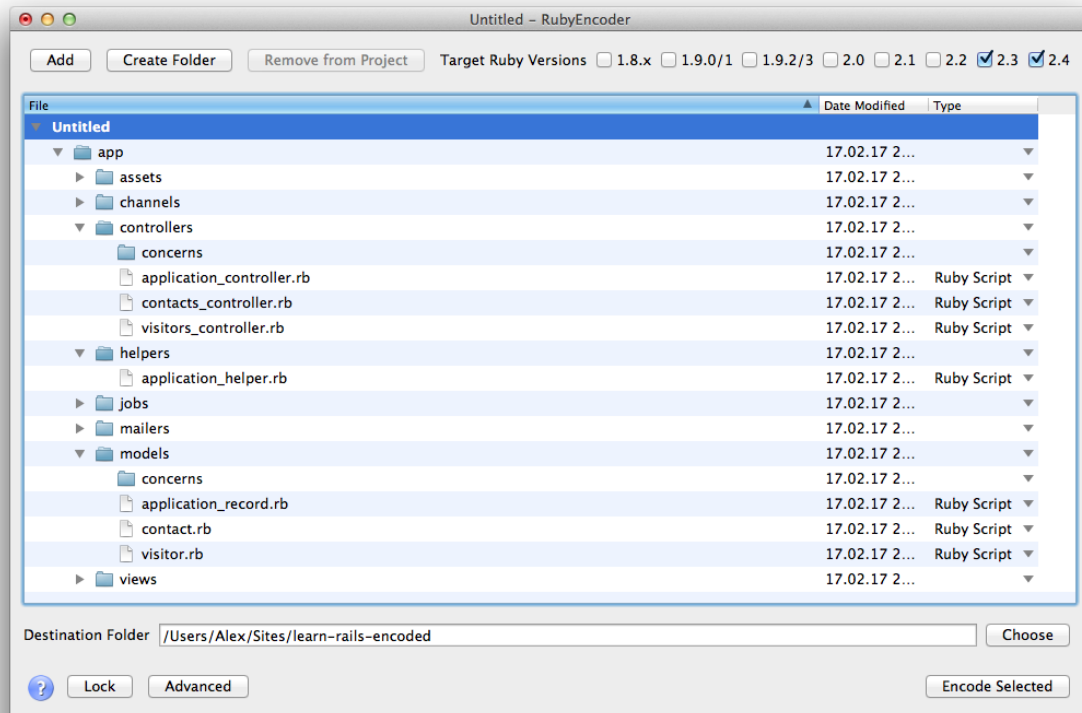
1. Start from making a full copy of your RoR application in the new folder. That new folder will be the destination folder for your encoded application.

You need to copy the application to the new folder because RoR application contains many other files beside those you encode. These files include non-Ruby files which you cannot encode with RubyEncoder, standard RoR Ruby files which you technically may encode but there is no reason for encoding standard RoR files as they are open source, configuration Ruby files from /config subfolder (which technically you may encode later if necessary, see below).

2. Run RubyEncoder and add your source /app folder to the project. Add source, not copied folders. Select target versions of PHP according to your version or versions of Ruby that runs your Rails application.

When adding the files, navigate using the tree view on the left hand side panel in the popup window, select folders using the right hand side panel.

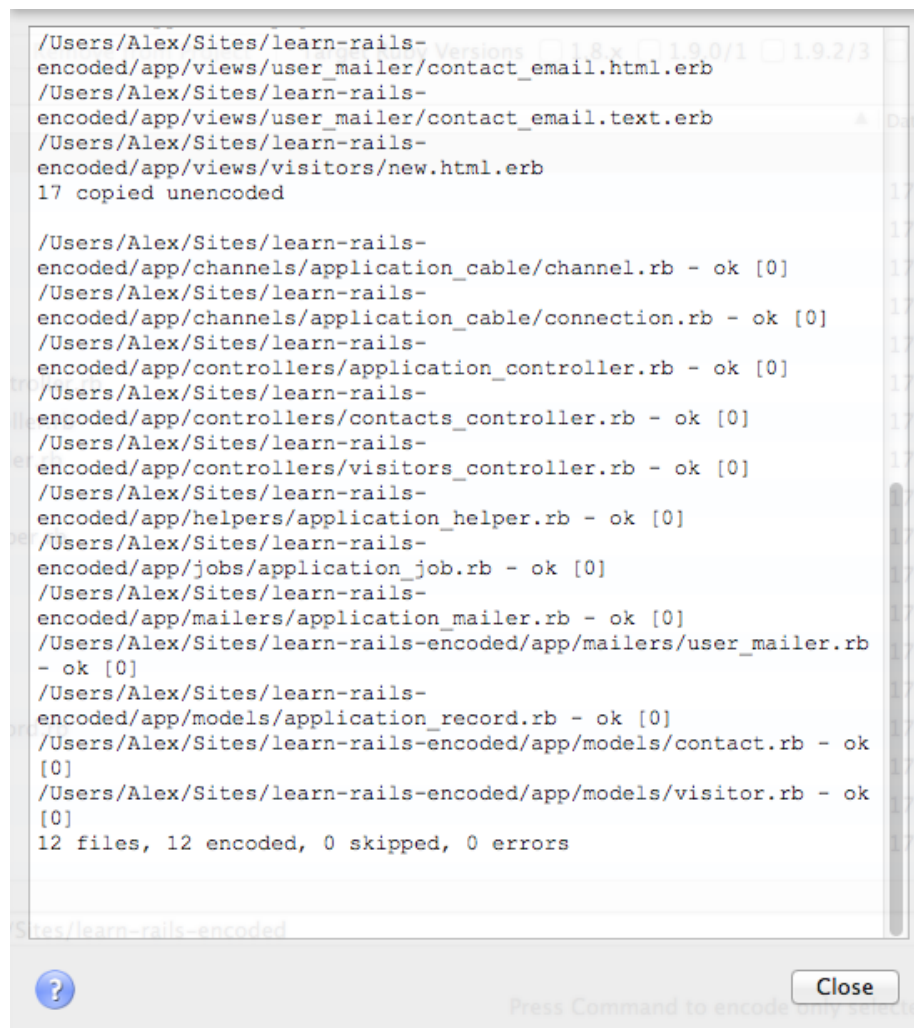
Please see the result on the screenshot below. Screenshots were taken on Mac (sorry Windows and Linux users) but it works and looks exactly the same on any platforms where you run RubyEncoder. File types will be automatically set as 'Ruby Script' for your *.rb files in the /app folder.



3. Select the destination folder by clicking 'Choose' button. Select the target folder you copied your original source to during the first stage above.

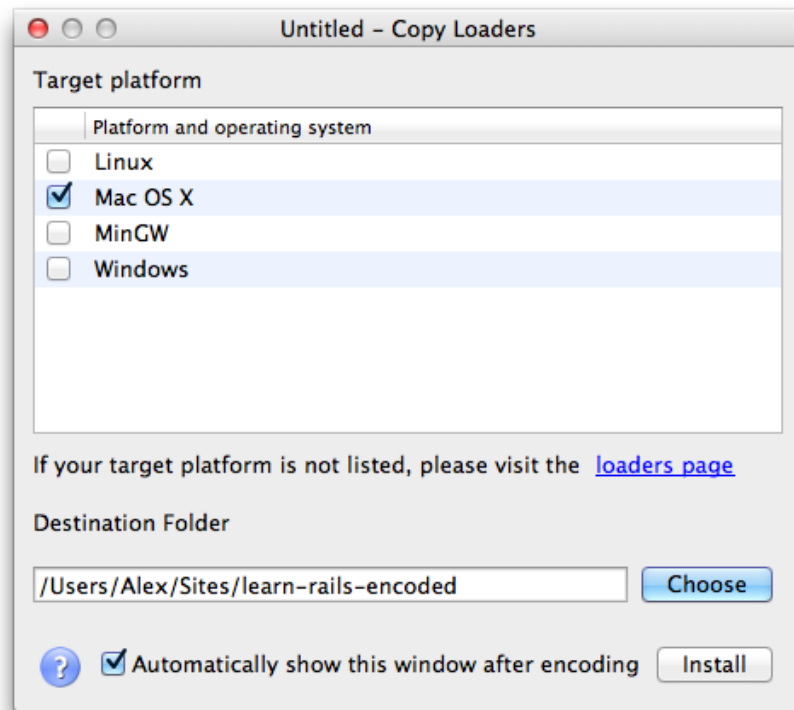
4. Click 'Encode'. Files you added to the project will be copied from the source to the destination folder (overwriting old files) and then will be encoded in the target folder, to keep your source intact. Results of encoding are shown in the popup window.

Note, you cannot encode any non-Ruby files with RubyEncoder. E.g. *.erb will not be encoded but will be copied to the destination.



5. The loaders installation window appears with the destination folder selected. Choose the target platform or platforms to copy the loaders, click 'Install'.

Loaders for selected platforms will be copied to the /rgloader folder within your target encoded application folder. Select the platforms that will run your encoded RoR application. Also you may add or update the loaders later.



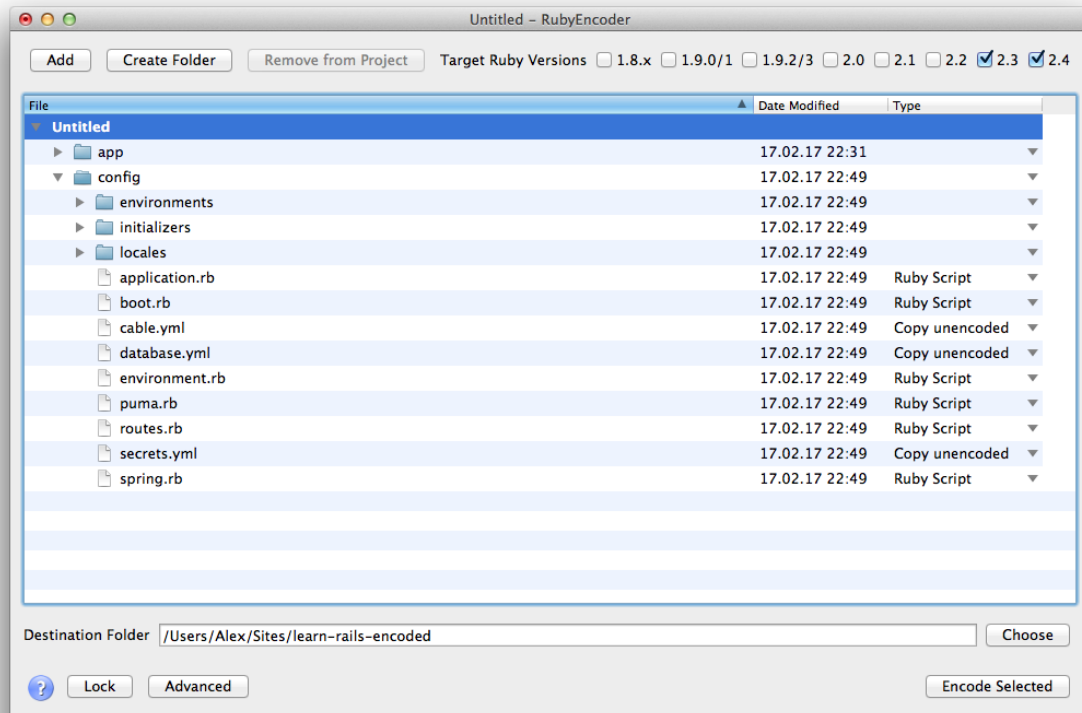
6) Open your favourite terminal and start the encoded Rails application from the destination folder selected in the above steps.

```
cd /path/to/your-encoded-application  
bin/rails server
```

Check results in your web browser as usual. If you have any questions, please don't hesitate to contact us support@rubyencoder.com

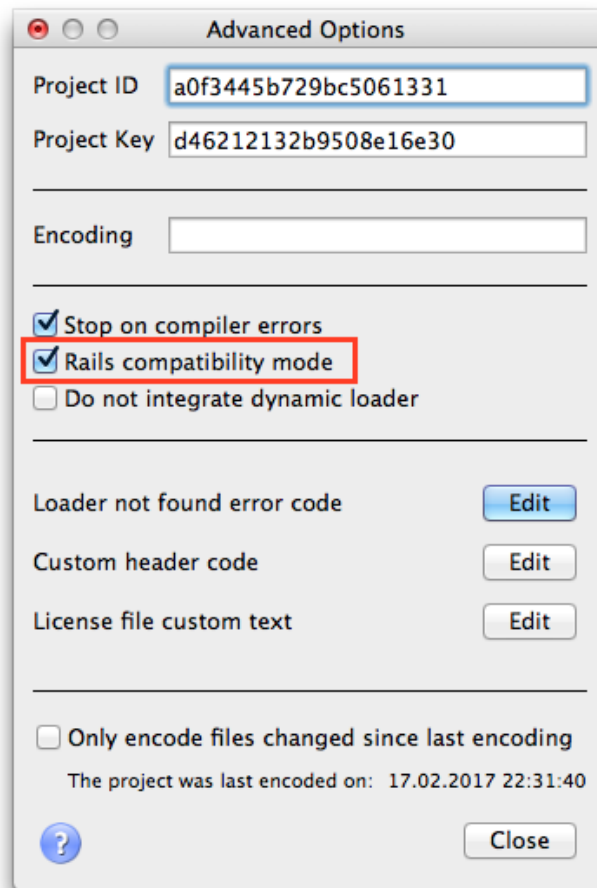
If you also need to encode your RoR application configuration files, those stored in /config subfolder, please read below. Note, you may encode only Ruby files, you can't encode say YAML database config (as it's not Ruby).

1) Select the root element in the project tree and click 'Add', select /config folder in your source RoR directory, add it to the project.



2) Open Advanced settings and select 'Rails compatibility mode'.

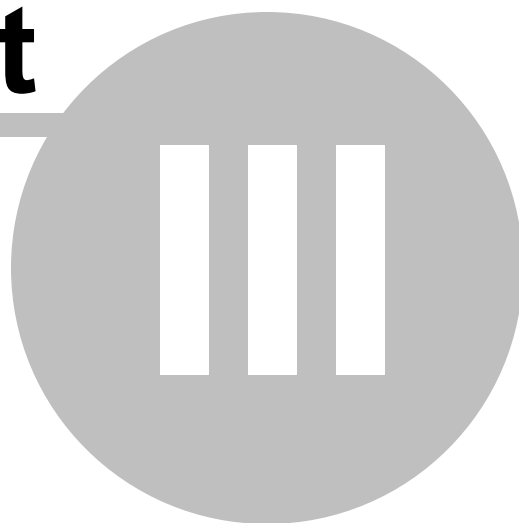
You may find more information about this option [in this topic](#).



3) Click 'Encode', install loaders if not installed yet, run the encoded application. Done.

RubyEncoder 3 **for Windows, MacOS and Linux**

Part



3 Command line encoder

3.1 Command line tools installation

You may download RubyEncoder 3 installation package as .zip, .tar.gz or tar.bz2 file or .dmg file for MacOS. Windows version includes an installer which you may run to install RubyEncoder. An evaluation version of RubyEncoder is available as a free download from our site <http://rubyencoder.com/trial.html>. A full version of RubyEncoder is available for download from your account profile after purchase.

[Click here](#) to read more about installation for Windows.

[Click here](#) to read more about installation for MacOS.

[Click here](#) to read more about installation for Linux.

[Click here](#) to read more about installation for FreeBSD.

[Click here](#) to read more about installation for Docker. For docker you need an appropriate version of RubyEncoder for Linux.

3.1.1 Windows

Installation for Windows is easy. Double click the downloaded RubyEncoder installer executable rubyencoder-3-windows.exe or rubyencoder-3-evaluation-windows.exe in order to run it. Follow instructions on the screen. RubyEncoder installer will install the software and then run the GUI encoder.

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

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

C:\Program Files\RubyEncoder 3\bin

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

3.1.2 MacOS

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

Command line encoder and tools are included with the GUI installation for MacOS. You may find them within RubyEncoder.app application package. Right click the RubyEncoder icon in Finder, choose 'Show

Package Contents', navigate to Contents/MacOS. Note, the command line encoder is named 'rgencoder' to distinguish it from the GUI application.

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

/Applications/RubyEncoder.app/Contents/MacOS/bin

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

3.1.3 Linux

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

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

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

RubyEncoder-3-Evaluation-Linux-x86_64.tar.gz
RubyEncoder-3-Evaluation-Linux-x86_64-CLI.tar.gz
RubyEncoder-3-Linux-x86_64.tar.gz
RubyEncoder-3-Linux-x86_64-CLI.tar.gz

Unpack the downloaded encoder installation file.

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

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

```
> cp /your/path/to/downloaded/RubyEncoder-3-Evaluation-Linux-x86_64.tar.gz /home/username
```

Unpack:

```
> cd /home/username  
> tar xzf RubyEncoder-3-Evaluation-Linux-x86_64.tar.gz  
  
> cd RubyEncoder*
```

Run the GUI encoder:

```
> ./RubyEncoder
```

Optionally install icons to your Linux desktop (GNOME):

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

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

Copy the downloaded file to the destination directory /home/username. You may use either OS interface

or terminal to locate and copy the downloaded file:

```
> cp /your/path/to/downloaded/RubyEncoder-3-Evaluation-Linux-x86_64-CLI.tar.gz /home/username
```

Unpack:

```
> cd /home/username
```

```
> tar xzf RubyEncoder-3-Evaluation-Linux-x86_64-CLI.tar.gz
```

```
> cd rubyencoder*/bin
```

Run the CLI encoder, proceed to automatic registration:

```
> ./rubyencoder
```

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

```
> ./rubyencoder
```

The CLI installation package has the following structure:

rubyencoder-3/bin/rubyencoder	RubyEncoder executable
rubyencoder-3/bin/rubyencoder1*.so	Internal encoder for Ruby 1.8,1.9 (cannot be used directly)
rubyencoder-3/bin/rubyencoder2*.so	Internal encoders for Ruby 2.x (cannot be used directly)
rubyencoder-3/bin/license.txt	license text
rubyencoder-3/bin/licgen	RubyEncoder Script License Generator (for full version only)
rubyencoder-3/bin/rginfo	RubyEncoder Information Tool (for full version only)
rubyencoder-3/rgloader/*	Protected script loaders
rubyencoder-3/README	Startup document
rubyencoder-3/User_Manual.pdf	User manual in PDF format

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

3.1.4 FreeBSD

You need to unpack the downloaded encoder installation file into any directory. For FreeBSD we recommend that you install to /usr/local or your home directory /usr/home/username. Only command line (CLI) interface is available for FreeBSD. The following instruction is expecting you use a terminal for installation. The downloaded installation package may have the following names according to the OS, platform and version of the encoder:

```
rubyencoder-3-freebsd-i386.tar.gz (or .zip, or .tar.bz2)
rubyencoder-3-evaluation-freebsd-i386.tar.gz (or .zip, or .tar.bz2)
rubyencoder-3-freebsd-x86_64.tar.gz (or .zip, or .tar.bz2)
rubyencoder-3-evaluation-freebsd-x86_64.tar.gz (or .zip, or .tar.bz2)
```

Example (note, tar.gz package file name may differ):

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

```
> cp /your/path/to/downloaded/rubyencoder-3-evaluation-freebsd-i386.tar.gz /usr/local
```

Unpack:

```
> cd /usr/local
> tar xzf rubyencoder-3-evaluation-freebsd-i386.tar.gz
```

Update permissions:

```
> cd /usr/local/rubyencoder-3-evaluation/bin
> chmod a+x rubyencoder
```

Run the CLI encoder, proceed to automatic registration:

```
> ./rubyencoder
```

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

```
> ./rubyencoder
```

The CLI installation package has the following structure:

rubyencoder-3/bin/rubyencoder	RubyEncoder executable
rubyencoder-3/bin/rubyencoder1*.so	Internal encoder for Ruby 1.8,1.9 (cannot be used directly)
rubyencoder-3/bin/rubyencoder2*.so	Internal encoders for Ruby 2.x (cannot be used directly)
rubyencoder-3/bin/license.txt	license text
rubyencoder-3/bin/licgen	RubyEncoder Script License Generator (for full version only)
rubyencoder-3/bin/rginfo	RubyEncoder Information Tool (for full version only)
rubyencoder-3/rgloader/*	Protected script loaders
rubyencoder-3/README	Startup document
rubyencoder-3/User_Manual.pdf	User manual in PDF format

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

3.1.5 Docker

Installing to a Docker container is experimental as of version 2.5 of RubyEncoder.

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

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

```
RubyEncoder-3-Evaluation-Linux-x86-CLI.tar.gz
RubyEncoder-3-Linux-x86-CLI.tar.gz
RubyEncoder-3-Evaluation-Linux-x86_64-CLI.tar.gz
RubyEncoder-3-Linux-x86_64-CLI.tar.gz
```

Please follow the instruction below in order to install RubyEncoder to a Docker container and then run it

from there for encoding your files as a part of your project deployment process. Note, every installation of RubyEncoder to another machine or another Docker machine still requires an additional license as one license lets you install and use RubyEncoder only on one machine.

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

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

```
> cp /your/path/to/downloaded/RubyEncoder-3-Linux-x86_64-CLI.tar.gz /home/username
> cd /home/username
> tar xzf RubyEncoder-3-Linux-x86_64-CLI.tar.gz
```

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

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

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

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

3) Copy RubyEncoder encoder binaries to the rubyencoder subfolder within the docker-installation

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

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

```
> vi entrypoint.sh
```

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

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

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

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

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

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

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

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

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

3.1.6 First run

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

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

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

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

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

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

3.2 Running the command line encoder

RubyEncoder 3 is a command line executable file named 'rubyencoder'. You may find it in the RubyEncoder installation directory in the `/bin` subdirectory. Running the encoder without any options prints a list of all available options for a quick help. Please refer to this user manual for details of using the encoder. In your terminal change the current directory to the RubyEncoder installation directory and type `bin/rubyencoder` to run the encoder.

3.2.1 Note for GUI users

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

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

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

3.2.2 Usage

Run the rubyencoder executable without parameters to get a list of all available options.

```
single file:  rubyencoder [options] file.rb
multiple files: rubyencoder [options] file1.rb file2.rb file3.rb
file mask:   rubyencoder [options] *.rb
file list:   rubyencoder [options] @filelist
```

You may run the RubyEncoder 3 encoder to encode either one or multiple files. You may enumerate all files you want to encode or use a file mask or file list to specify multiple files. A file list is a text file with either full or relative file paths of all the files to encode, separated by a new line. You should use an @ sign before the filelist name in the command line. A file list passed to the RubyEncoder encoder for batch processing from the command line may contain file masks. Standard ? and * symbols are available.

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

If -o option is used to specify output directory the original file will not be backed up. Instead of it, the original file will be copied to the output directory and encoded there. We recommend that you always encode to the output folder specified with -o.

--ruby <version x.y>

RubyEncoder CLI encoder encodes scripts for Ruby 1.9.x and 2.x by default. Use --ruby option to specify versions of Ruby you need to encode scripts for. Available values for --ruby option are 1.8, 1.9 (encoding for 1.9.0/1.9.1), 1.9.2 (encoding for 1.9.2/1.9.3), 2.0.0, 2.1.0, 2.2.0, 2.3.0, 2.4.0, 2.5.0 etc. To encode a script to run under Ruby 1.8.x, 1.9.x and/or 2.x use the --ruby option multiple times and specify all versions of Ruby you need to run protected files under.

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

Example:

```
> rubyencoder file1.rb
> rubyencoder --ruby 2.4.0 --ruby 2.5.0 file2.rb file3.rb file4.rb
```

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

--ruby 2.4.0+ RubyEncoder)	encodes for Ruby 2.4 and all the newer versions (up to 2.5 for this version of
--ruby 2.0.0+ RubyEncoder)	encodes for Ruby 2.0 to 2.5 and newer (up to 2.5 for this version of
--ruby 2.3.0-	encodes for Ruby 2.4 and newer, i.e. excludes Ruby 2.3 and older

You may enumerate all the files you want to encode in a file list. A file list is a text file with either full or relative file paths of all the files to encode, separated by a new line (masks are supported, use "*" and "?" for it). You should use an @ sign before the file list name in the command line.

Usage: >rubyencoder @filelist

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

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

>rubyencoder -r source_dir

which will work as if a file mask was specified:

>rubyencoder -r "source_dir/*"

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

ok	The file was encoded without problem.
file not found cannot be read	The specified file could not be found. Check the specified file path.
Ruby syntax or other compiler error	The original file has syntax or other errors and thus cannot be encoded. Check your file, test it with the Ruby interpreter. This error usually appear when encoding for multiple versions of Ruby and if your file is not compatible with all the target versions of Ruby. Make sure your ruby script is compatible with all versions of Ruby you are encoding for.
could not backup source file, skipped	The encoder could not make a backup copy of your original file (when no output directory was specified). RubyEncoder skips the file in that case to keep your original version. Check you have enough free space available and permissions to write to your original files directory.
cannot not write file	The encoder could not write the encoded file. Check you have enough free space available and permissions to write to original files directory or to the output directory if you have specified it with the -o option.
file is already processed by RubyEncoder	The encoder will not encode files which are already encoded with RubyEncoder. Check your original files directory.
empty file, skipped	The encoder will not encode empty files. If you need to have empty files for any reasons you may copy them manually.

not regular file, skipped	The encoder could not encode a file because it is not a regular file. It may be a socket or a unix device for example.
do not encode, skipped	The file was marked as 'do not encode' and therefore was skipped.
copied	The file was copied without encoding. It is possible when -f option is used to specify files to encode and -o option is used to specify output directory. All other files than specified in -f option will be copied as-is without encoding. It is useful for encoding an entire project directory when it also may contain non-Ruby files.
internal encoder error, unknown error	This is an internal problem with the encoder. Check you have enough free memory space to run the encoder and some free space on the disk. If it is not a memory problem then let us know about this error. Send an email to support@rubyencoder.com with a detailed description of the error and the command line used for running the encoder. We will investigate the problem. We may also need some additional information from you.

3.2.3 Output directory for encoded scripts

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

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

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

Command line option: -o <output_dir>

Example 1: Encode all *.rb scripts in the current directory with recursion and put encoded files to /home/myproject/encoded.

```
>rubyencoder -r -o /home/myproject/encoded "*.rb"
```

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

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

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

3.2.4 Specifying which files to encode and which to copy_2

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

specified files will be encoded by default.

Example 1:

```
>rubyencoder -r -f "*.rb" -o "output_dir" "**"
```

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

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

Example 2:

```
>rubyencoder -r -f "*.rb" -f "includes/*.inc" -f @myrubyfiles -o "output_dir" "**"
```

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

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

E.g. you may encode *.rb files but keep *.erb or config.rb unencoded and copy the latter ones as-is:

```
>rubyencoder -o /path/to/target -r -f "*.rb" -c "*.erb" -c "config.rb" /path/to/source
```

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

3.2.5 Excluding files from processing

You may exclude some files or directories from processing. Please use --exclude=mask option to specify file(s) and/or dir(s) to exclude from processing. You may specify either a strict name, relative path with a directory name or a mask (with ? and/or * wildcard symbols). Wildcards in directory names are also supported. Note, excluded files will neither be processed, nor be copied to the target (-o) folder.

Example: `rubyencoder -r --exclude "doc/*" --exclude "config.rb" "*.rb"`

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

You may enumerate all the files you want to exclude from encoding using a file list to specify multiple files. A file list is a text file with either full or relative file paths of all the files to exclude, separated by a new line (masks are supported, use * and/or ? wildcard symbols). Specify the @ before the filelist name in the command line, e.g. -x @excludefilelist

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

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

3.2.6 Encoding entire directory contents_2

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

```
>rubyencoder -r source_dir
```

instead of the following

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

3.2.7 Locking options (full version only)

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

--expire <dd/mm/yyyy>

Using this option you can set an expiration date for the script. The script will NOT run on and after the specified date and comes with the error message: "script has expired".

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

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

Example 1: --expire 10d

Example 2: --expire 1d12h

Example 3: --expire 90m

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

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

NTP protocol is used, UDP on port 123 on remote server. NTP is always tried at first. TIME protocol is the second, TCP on port 37 on remote server, checked if no response on NTP port.

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

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

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

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

A list of available time servers may be found [here](#).

Locking the script to work only online

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

--domain <domain>

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

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

Please note, the loader is able to check the domain name ONLY if the protected script is running in a web server environment. The web server must also provide the Ruby interpreter with one of the following environment variables; SERVER_NAME or HTTP_HOST. The Apache web server does it by default for CGI interface. It also provides these environment variables for FastCGI but then it depends on the FastCGI server's resending these variables to the script. For the Nginx webserver you need to have an appropriate fastcgi_param line in the web server configuration file to have that variable passed to the Ruby script. For other web servers please find and use appropriate configuration directives.

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

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

Example 1: --domain mydomain.com

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

Example 2: --domain www.mydomain.com

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

--domain-encrypt <domain>

Bind and encrypt the script with a domain name. The encoder will lock the script to run only from the specified domain. The encoder will also use a specified domain name as a part of the key for encryption for enhanced security. The loader will not be able to decrypt a script in it's run from the invalid domain

and then the script fails with the error message: "RubyEncoder Loader - Loader - script checksum error. The encoded file has been modified. If the script requires a license file to run this error may be caused by an invalid license file. Install the original unmodified file or contact the script author about getting the original file or license file. Error code [12]". You may use this option ONLY ONCE in the command line. This option may not be used along with the other --domain options.

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

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

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

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

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

Please note, the loader is able to check the domain name ONLY if the protected script is running in a web server environment. The web server must also provide the Ruby interpreter with one of the following environment variables; SERVER_ADDR or LOCAL_ADDR. The Apache web server does it by default for CGI interface. It also provides these environment variables for FastCGI but then it depends on the FastCGI server's resending these variables to the script. For the Nginx webserver you need to have an appropriate fastcgi_param line in the web server configuration file to have that variable passed to the Ruby script. For other web servers please find and use appropriate configuration directives.

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

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

Bind and encrypt to IP/mask. The encoder will lock the script to run only from the specified IP address. The encoder will also use the specified IP address with applied mask as a part of the key for encryption for enhanced security. The Loader will not be able to decrypt the script running from the invalid IP address and will fail with the error message: "RubyEncoder Loader - Loader - script checksum error. The encoded file has been modified. If the script requires a license file to run this error may be caused by an invalid license file. Install the original unmodified file or contact the script author regarding getting the original file or license file. Error code [12]". You may use this option ONLY ONCE in the command line. IP address mask 255.255.255.255 is used by default if not specified. This option may not be used along with other --ip options.

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

--mac <XX:XX:XX:XX:XX:XX>

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

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

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

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

--machine-id <machine id>

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

Use RGLoader::get_machine_id() API method and get the machine ID of the computer where you run this method. It may be called directly or from the encoded Ruby code (not locked with any options). Note, as this API method is binary compiled into the loader, an appropriate RubyEncoder loader but me installed and loaded (require directive) before your code calls this method. We recommend that you create a mini project, encode it and include the loaders for obtaining the machine ID from the client's machine before locking to it, in that case loaders will be found automatically.

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

--machine-id-encrypt <machine id>

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

--external <filename>

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

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

The algorithm which is used for locking scripts to an external license gives your scripts much stronger protection from reverse engineering, unlocking and bytecode stealing, but it also gives you the most flexible way to generate trial versions of your products and to lock scripts to your customer's machine. This is the most powerful and flexible way to protect your scripts. We recommend that you use external license file locking for all your scripts. Using of locking to a license file is also the best way of deploying the encoded project to different clients as you may encode once and then assign different restriction options for the clients in the licenses you generate per client.

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

--projid <value>

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

--projkey <value>

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

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

--remote-verification-url <URL>

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

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

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

```
print RGLoader::get_verification_id()
```

Note 1: if you use any frameworks, you may need to use another mechanism for sending a string to the output stream instead of `print`,
e.g. *res.write* for Cuba etc

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

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

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

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

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

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

Note 3: you should not worry about the delays HTTP adds, they are minimal as your CLI code and the

main WEB code are both running on the same machine. However, it's recommended to check the verification URL is accessible from CLI, e.g. using 'curl' or 'wget'. If the domain name can't be recognised from CLI, you may add the domain name to hosts file or switch to using IP instead of the domain name.

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

Important Security Notice!

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

(!) Remember your Project Id and Project Key. It's impossible to restore the values if forgotten. They are required for generating licenses for your customers.

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

3.2.8 Advanced options

--encoding <encoding>

Specify a target character encoding for ruby 1.9.x and 2.x code. This option does not make any sense for Ruby 1.8. RubyEncoder compiles source Ruby files to a binary representation, as a result 'magic comments' cannot be used for specifying character encoding of source files as they are read on the 'running' stage, not during compilation. Please use this RubyEncoder option when you are encoding files that have 'magic comments' in the code for specifying the character encoding.

Example 1: --encoding UTF-8

Example 2: --encoding ISO-8859-1

Since version 2.5 of RubyEncoder the default encoding is UTF-8 if not specified.

--rails

Enables Rails compatibility which lets you encode all Rails *.rb files (you can encode only pure Ruby files with RubyEncoder). Normally you can encode only application controllers, model and helper files. Other files if encoded would not work under Rails if the Rails compatibility mode was not used. Using this option makes an internal CRC check weaker for a bit. This is done to let Rails engine use eval() for loading protected files.

-n

Don't integrate dynamic loader. You may use this option if you don't want to include the default starter code into protected scripts. Scripts encoded using this option will not be able to automatically find and load an appropriate RubyEncoder loader and you have to start an appropriate loader before running the encoded script. This option is useful if you have multiple encoded files and want to start the

RubyEncoder loader manually from your code maybe from the custom folder before running the protected files. When starting loaders manually you may either 'require' the loader.rb helper file which is available along with binary loaders or you may load the binary loader directly.

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

-p "code"

Prepend header code. You may put any code to be inserted BEFORE the protected scripts code. This code WILL NOT BE ENCODED. This should be syntactically correct Ruby code. Of course, it may by Ruby comments starting with a # symbol. This option is usually used for including copyrights into protected scripts. Also this option is used to put a custom error handler code into protected scripts (see --catch option). You should prepend all special characters including double quote characters with a back slash if you want to include them into the code (" becomes \ ").

Example 1:

```
>rubyencoder -p "# My protected script. Copyright by My Name" file.rb
```

Example 2:

```
> rubyencoder -p "puts \"My protected script. Copyright by My Name\";" file.rb
```

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

- {RG_DATE} - current date i.e. date of encoding
- {RG_LICENSEE} - RubyEncoder license owner from the RubyEncoder license file

It works in the same way also for licgen and do replacements for custom text if it is used. [See details](#)

-j "code"

By default a RubyEncoder protected script will generate an exception with the following message when the RubyEncoder Loader is not installed:

Ruby script '...' is protected by RubyEncoder and requires the RubyEncoder loader. Please visit the <http://www.rubyencoder.com/loaders/> RubyEncoder site to download the required loader and unpack it into '.../rgloader/' directory to run this protected script. (RuntimeError)

It is possible for you to change the default loader error behavior. This option allows you to change the default error action of the protected script if it cannot find or load an appropriate RubyEncoder Loader file. You may use any Ruby code here and it will be executed as a replacement to the default RubyEncoder loader exception. This code WILL NOT BE ENCODED. You should prepend all all special characters including double quote characters with a back slash if you want to include them into the code (" becomes \ "). If you want you may load the required RubyEncoder Loader with "require" directive from a non-standard directory instead of printing an error message.

Example:

```
> rubyencoder -j "puts \"Loader is not found. Call 123-456-7890 for support\"; exit(1);" hello.rb
```

The encoded script will have the defined code as unencoded:

```
# RubyEncoder v1.0

_d = _d0 = File.expand_path(File.dirname(__FILE__)); while 1 do _f = _d + '/rgloader/
loader.rb'; bre
ak if File.exist?(_f); _d1 = File.dirname(_d); if _d1 == _d then

puts "Loader is not found. Call 123-456-7890 for support"; exit(1);

break; else _d = _d1; end; end; require _f; RGLoader::load
('AAEAAAAEaAAAAIAAAAA/6vBQ2j8ivZCR3gVFB15
Hr/Y90fXBYLdnQDavaJb9qL66uNG0QBaN4Uk9NrQubHzhv/
WHM97yUQkyjHsid9nsMr9JiGEavcQ5tAiBHD1/lxoxia2TOrPle4V
e8
+H+3odwWqOIjks94QLEgAAAGAAAAB75w3qzOZQcmvQDuOs+G9ZVhz0GbAy1oT4injtT83Ii jyj0iubOUfKRn2
+frb7QOm3dEXG
qgUtKkGzz2yaCE0T9yV1V0ZtZv4RKezGylUJdYtDb4lrK0t8S9FRLBYnwAYAAAAA');;
```

Now a test run without a required RubyEncoder Loader installed:

```
>ruby hello.rb
```

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

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

Prepend header code option -p and Loader error code option -j may load the source from a file. Use @filename as a parameter for -p or -j. We suggest to use this option if you need to add some complex code. It is easier to edit this code in a separate file than writing it in command line as an option.

Example:

```
>rubyencoder -p @prepend.rb -j @loadererr.rb file.rb
```

--stop-on-error

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

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

-a <YYYYMMDDhhmmss>

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

3.2.9 Custom predefined constants

--const name=value

RubyEncoder lets you define custom named constants during the encoding process, or within the external script license. Constant name/value pairs are stored internally in the encrypted area of the protected script or the license. They may be used for custom script locking or any other actions if you need to store a custom value in the protected script or script license file and then read it from your protected Ruby code. Note, the values are store in the encrypted area and can't be modified by the users.

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

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

```
>rubyencoder --const "licensed_for=Robin Hood" myscript.rb
>licgen --const "licensed_for=Robin Hood" script.lic
```

To get a predefined constant value from the protected Ruby code use RGLoader::get_const() method. This method is defined in the RubyEncoder loader.

RGLoader::get_const() will return a predefined RubyEncoder constant value or nil if the constant with the specified name is not defined. Constant names are **case sensitive**.

Note, as RGLoader::get_const() is a part of the loader API, it may be used only from protected files.

There are 5 predefined constants for any protected script:

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

3.2.10 Custom error handling

--catch err=function

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

```
error_handler( code , message )
```

You may use any name for this function. Also you may have different functions for different script errors. The first argument is the error code. The second one is the default error message. To set a custom error handler use `--catch` option for the `rubyencoder` command:

```
>rubyencoder --catch err=function myscript.rb
```

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

Err tag	Returned code	Default message
ERR_LICENSE	01,02,03, 04,05	script cannot run on this machine (code indicates a reason: 01-IP, 02-domain, 03-MAC address, 04-machine ID, 05-remote verification URL locking)
ERR_EXTLICCRC	06	script license is invalid
ERR_EXPIRED	09	script has expired
ERR_EXTLIC	13	script requires license file to run
ERR_OFFLINE	20	script requires an internet connection to run
ERR_ALL	-	-

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

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

Example:

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

```
>rubyencoder -p "def my_err_handler(code,msg); printf \"My error handler caught error code %d with a message '%s'\n\", code, msg; end;" --catch ERR_ALL=my_err_handler --external script.lic --projid Fg3161jd --projkey 826Gdb31 hello.rb
```

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

```
# RubyEncoder v1.0

def my_err_handler(code,msg); printf "My error handler caught error code %d with a message '%s'\n", code, msg; end;

_d = _d0 = File.expand_path(File.dirname(__FILE__)); while 1 do _f = _d + '/rgloader/loader.rb'; break if File.exist?(_f); _d1 = File.dirname(_d); if _d1 == _d then raise "Ruby script '"+__FILE__+"' is not protected by RubyEncoder and requires the RubyEncoder loader. Please visit the http://www.rubyencoder.com/loaders/ RubyEncoder site to download the required loader and unpack it into
```

```
'"+_d0+"/rgload
er/' directory to run this protected script."; break; else _d = _dl; end; end;
require _f; RGLoader:
:load( 'AAEAAAAAEoAAAAIAAAAAA/y7C/
NTZP8FnCp00d+uHwMSVgcicoaW6ERaCNJzTN2xah6H+6o9f2eGRGZuRBc9AjPFIowkPt
+ZUC2o6qTmfb4Dk6gJ30sVgI20+Yzju02WIv3pGK3UDOMw+MFuXAIp7
+8AL1Yy4FLawYTxZqzQTnsCQgOFObFkJhkIdkqAHHKdyf
Y7Fy0N1IhlqOMT1AWwOan2WIpmKbwbxl3vykcvUpncSAAAYAAAAAPwjwCB2fc7tjHyItPnulhcuoPtNMLr5l
XrpsB9d1KJVC0+W
IrJB4Y30lHCvzSDsC7dJ/Ak5LMBV/fVHQgOV2Is3B2/
SgalIdFc1lJ6ImqOJGBQDpAjBya977eTnLMDwAAAAA=' );
```

Now a test run without the required license file:

```
>ruby hello.rb
```

```
My error handler caught error code 9 with a message 'RubyEncoder Loader - This script
has expired. Contact the script author about this problem. Error code [09]'
```

3.2.11 Other options

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

-v	Display version number
-h	Display full options list
--license	Display license information
--license--release	Release the current license which allows reinstalling to another machine or the same one. You get 3 free license resets for every license.
--credits	Display names of RubyEncoder developers
-w	Wait for key press before exit. Allows you to check encoding log in terminal before exit.
-q	Display settings and request confirmation. The encoder will display all encoding parameters and wait for a key press before any real encoding takes place. You may check all parameters and cancel if anything is not correct.
--verbose <n>	Controls the encoder output. 0-quiet, 1-print only errors to the log, 2-print standard log. 2 is default
-r{n}	Recurse subdirectories. The encoder will process all subdirectories recursively when searching files using specified file masks. {n} is an optional directory trimming level, see below. IT IS IMPORTANT TO ENCLOSE FILE MASKS IN DOUBLE QUOTES. Otherwise, if file masks are not be enclosed in double quotes command line shell will expand file masks

and recursion will not work as expected.

-b <ext> Set file extension for backup files (bak is default). You may change an extension used for backup copies with this option.
Example: -b old

-b- Disable backup of source files (BE CAREFUL!)

-x "mask" | @list You may specify what files or directories shall NOT be encoded. You may specify either a strict name, relative path with a directory name or a mask (with ? and/or *).

Example:

```
>rubyencoder -r -x "doc/*" -x "config.rb" "*.rb"
```

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

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

When specifying a relative path don't use ../ or ./ directory specifiers.

-f "mask" | @list You may specify what files will be encoded by filenames, file masks or a file list. All other files which have been added for processing or found by expanding file masks will be copied into the output directory "as-is" without encoding. If you don't specify the -f option then all specified files will be encoded by default.

Example 1:

```
>rubyencoder -r -f "*.rb" -o "output_dir" ""
```

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

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

Example 2:

```
>rubyencoder -r -f "*.rb" -f "includes/*.rb" -f @myrubyfiles -o "output_dir" ""
```

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

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

When specifying a relative path don't use ../ or ./ directory specifiers.

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

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

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

```
> rubyencoder -r -o /home/myproject/encoded *.rb
```

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

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

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

Recurring subdirectories

You may use wildcards in source directory names, e.g. /path/to/dir?/*.*rb This also works in @filelist and you may use it with -f, -c, -x. If the @filelist is specified as source, recursion is automatically turned on, but it's still possible to change the directory trimming level with -r{n} if necessary.

Optional directory trimming

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

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

```
/source/file0.rb  
/source/dir1/file1.rb
```

```
/source/dir2/file21.rb  
/source/dir2/file22.rb
```

and encoding to the /target with the following command

```
rgencoder -o /target -r /source
```

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

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

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

```
rgencoder -o /target -r1 /source
```

and get the following file structure in the target

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

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

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

Encoding with the following command in default mode works well

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

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

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

one would expect

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

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

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

3.2.12 Encoding to standard output

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

For example:

```
>rubyencoder --ruby 2.0.0 -- < /path/to/source.rb > /path/to/encoded.rb
```

3.2.13 Exit codes

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

Exit code	Reason
0	no error
1	file not found
2	ruby syntax or other compiler error
3	could not backup a file when backup is on
4	could not write output file
5	file is already encoded
6	license error
7	license error
8	usage error, check command line options
9	cancelled, no error but files were not encoded, e.g. help screen shown or license information
10	license expired (for trial version)
11	empty file, skipped
12	not a regular file, skipped
13	file copied without encoding
14	encoded in template mode
15	file skipped
18	multiple files are being encoded and an error happens during encoding (see encoding log for

details)

255 other, internal or unexpected errors

3.3 Script license generator (full version)

The Script License Generator is an external tool for creating script license files. A script license file is required to run protected scripts encoded with the `--external` option. You may find 'licgen' executable in the RubyEncoder installation directory in `/bin` subdirectory. Running the license generator without any options prints a list of all available options for a quick help. Please refer to this user manual for details of using the license generator.

Using the script license is the best way of encoding if you need to distribute one script or entire project between different users but need to use different restriction options for each user. You need to encode your scripts with the `--external` option using RubyEncoder 3 and then create a license for each user with the RubyEncoder 3 Script License Generator.

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

If a protected script cannot find the specified license file it will return an error message: "RubyEncoder Loader - the script requires ... license file to run. Contact the script author regarding getting a license file. Error code [13]"

The algorithm used for locking scripts to an external license file gives your scripts much stronger protection from reverse engineering, unlocking and bytecode stealing, but it also gives you the most flexible way to generate trial versions of your products and to lock scripts to your customer's machine. This is the most powerful and flexible way to protect your scripts. We recommend that you use external license files for all your script protection.

A brief description of the algorithm

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

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

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

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

Command line example:

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

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

If a script is run with an incorrect license file the following error message appears:
"RubyEncoder Loader - a license file required to run this protected script is invalid. Contact the script author regarding getting a license file. Error code [06]"

If a script is run with a license file with the correct Project Id but incorrect Project Key (this may be a cracking attempt or accidental modification of the license file or script) the following error message appears:
"RubyEncoder Loader - Loader - script checksum error. The encoded file has been modified. If the script requires a license file to run this error may be caused by invalid license file. Install original unmodified file or contact the script author regarding getting the original file or license file. Error code [12]"

Important Security Notice!

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

(!) Remember your Project Id and Project Key. It's impossible to restore the values if forgotten. They are required for generating licenses for your customers.

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

3.3.1 Usage

licgen [options] output.lic

--expire <dd/mm/yyyy>	Set license expiration date
--expire <00d[00h[00m[00s]]>	Set license expiration time from now
--domain <domain>	Bind license to domain name
--ip <x.x.x.x[/y.y.y.y]>	Bind license to ip/mask
--mac <x:x:x:x:x:x>	Bind license to mac address
--machine-id <machine id>	Bind license to a machine id
--projid <value>	Set project id (required, the same as for encoding)
--projkey <value>	Set project key (required, the same as for encoding)
--const name=value	Set custom defined constant
--time-server <server,server,...>	Set time server (for expiration date check)
--text "text"[@file	Add plain text into the license file
-l	Display RubyEncoder license information for the tool itself
-w	Wait for key press before exit
-v	Display version number
-h	Display options help

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

external option during the encode.

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

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

Locking options

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

Custom constants

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

Options unique to the license generator

--text "text"

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

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

- {RG_DATE} - current date i.e. date of encoding
- {RG_LICENSEE} - RubyEncoder license owner from the RubyEncoder license file (i.e. your name, not your customer's name)
- {RG_EXPIRY_DATE} - your custom license expiry date. Changing of this text in the license file does not make sense and is NOT opening a back door, if you use the expiry date in your custom licenses. The expiry date is stored encrypted as well as other options within the license file. The readable text is only for information and you may put a name of your product, name of your customer and the expiry date.

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

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

3.4 File information tool (full version)

The Information Tool is an external tool for viewing protected scripts and script license files details. You may find the 'rginfo' executable in the RubyEncoder installation directory in /bin subdirectory. Running the information tool without any options prints a list of all available options for a quick help. Please refer to this user manual for details on using the information tool.

You may get information about protected scripts, or an external script license. This may be useful for supporting your customers, checking scripts or licenses passed to them etc. You may know the encode date, expiration date, locking options etc that were used during file encoding or script license generation. You may pass the encoded script name or script license as a parameter to this tool.

Script information: rginfo [options] file.rb

License information: rginfo [options] file.lic

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

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

3.4.1 Usage

Script information: rginfo [options] file.rb

Options:

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

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

License information: rginfo [options] file.lic

Options:

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

Other options:

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

RubyEncoder 3 **for Windows, MacOS and Linux**

Part



IV

4 Running protected scripts

Protected script loaders are dynamically loaded binary Ruby modules which are automatically used by the protected script for decrypting it, validating and then running the bytecode. The source code is never restored at any time, even in memory. There is no Ruby source code within the scripts protected with RubyEncoder. There are different versions of the loaders available for different operating systems and Ruby versions. The appropriate version of the loader will be automatically loaded by the protected script.

We periodically update RubyEncoder Loaders and add support for other operating systems. The latest loaders are always freely available from our site. If you have any problems with using the loaders please make sure you always use the latest version.

<http://www.rubyencoder.com/loaders/>

RubyEncoder Loaders are included with the RubyEncoder installation package. You may find the loaders in the /rgloader subdirectory within the RubyEncoder main installation directory.

4.1 Installing loaders

Scripts protected with RubyEncoder require installing a RubyEncoder Loader to the target machine in order to run. Protected scripts attempts to find the loader in the rgloader/ directory located within the protected script's directory or parent directories. The code will search for the rgloader/loader.rb helper script in it until it reaches the root / folder.

To run protected scripts on the target machine you need to copy the rgloader/ directory with its full content from the RubyEncoder main installation directory to the protected script directory on the target machine. If your project consists of multiple directories, with protected scripts in them, you may copy the rgloader/ directory into your project's top directory - protected scripts will be able to find the loaders there.

If you use an FTP client for file transfer make sure you use FTP BINARY mode for uploading loader files to the target machine.

It is not required to upload loaders for all supported platforms. You may upload loader.rb helper script and one required loader or only loaders for the target platform. See the [next section](#) for details about naming the loader files.

GUI has a built-in 'Loaders Installer' which you may find in the File menu.

4.2 rgloader directory structure

The following provides an overview of the rgloader/ directory which is a required location for RubyEncoder Loaders and loaders file naming conventions:

loader.rb	This file is a Ruby script which helps to automatically identify the required loader. This is a required file for installation on a target machine where protected ruby scripts will run. It is open and not encoded but you do not need to change it.
rgloader.darwin.bundle	Binary RubyEncoder Loader for Ruby 1.8.x for MacOS platform. It is a universal binary file compatible with i386 and x84_64 architectures.
rgloader19.darwin.bundle	Binary RubyEncoder Loader for Ruby 1.9.0/1.9.1 for MacOS platform.

	It is a universal binary file compatible with i386 and x84_64 architectures.
rgloader192.darwin.bundle	Binary RubyEncoder Loader for Ruby 1.9.2 for MacOS platform. It is a universal binary file compatible with i386 and x84_64 architectures.
rgloader.linux.so	Binary RubyEncoder Loader for Ruby 1.8.x for Linux i386 platform. It is 32-bit ELF shared object.
rgloader19.linux.so	Binary RubyEncoder Loader for Ruby 1.9.0/1.9.1 for Linux i386 platform. It is 32-bit ELF shared object.
rgloader192.linux.so	Binary RubyEncoder Loader for Ruby 1.9.2 for Linux i386 platform. It is 32-bit ELF shared object.
rgloader.linux.x86_64.so	Binary RubyEncoder Loader for Ruby 1.8.x for Linux x86_64 platform. It is 64-bit ELF shared object.
rgloader19.linux.x86_64.so	Binary RubyEncoder Loader for Ruby 1.9.0/1.9.1 for Linux x86_64 platform. It is 64-bit ELF shared object.
rgloader192.linux.x86_64.so	Binary RubyEncoder Loader for Ruby 1.9.2 for Linux x86_64 platform. It is 64-bit ELF shared object.
rgloader.freebsd.so	Binary RubyEncoder Loader for Ruby 1.8.x for FreeBSD 8.x+ i386 platform. It is 32-bit ELF shared object.
rgloader19.freebsd.so	Binary RubyEncoder Loader for Ruby 1.9.0/1.9.1 for FreeBSD 8.x+ i386 platform. It is 32-bit ELF shared object.
rgloader192.freebsd.so	Binary RubyEncoder Loader for Ruby 1.9.2 for FreeBSD 8.x+ i386 platform. It is 32-bit ELF shared object.
rgloader.freebsd.x86_64.so	Binary RubyEncoder Loader for Ruby 1.8.x for FreeBSD 8.x+ x86_64 platform. It is 64-bit ELF shared object.
rgloader19.freebsd.x86_64.so	Binary RubyEncoder Loader for Ruby 1.9.0/1.9.1 for FreeBSD 8.x+ x86_64 platform. It is 64-bit ELF shared object.
rgloader192.freebsd.x86_64.so	Binary RubyEncoder Loader for Ruby 1.9.2 for FreeBSD 8.x+ x86_64 platform. It is 64-bit ELF shared object.

The above list is not complete. Some of the new loaders for other operating systems and new versions of Ruby are probably already included into the installation package. We will include more loaders later to support new operating systems and platforms. Visit <http://www.rubyencoder.com/loaders/> for an update.

RubyEncoder 3 **for Windows, MacOS and Linux**

Part



V

5 Common mistakes

This section includes common mistakes that people may do, either in encoding and protecting their files, or in uploading or running them. They are not in any particular order, but we suggest that you take a look at this section before you contact us regarding any support matter.

5.1 Encoded scripts modification

Encoded scripts are protected against modification. Please **DO NOT MODIFY** any single byte in the encoded scripts or you will get an error when running them. While normally you will get an error message from RubyEncoder loader saying about a CRC error if running the modified protected script, running modified protected scripts may cause serious problems like segmentation faults in the Ruby interpreter. This is not a problem with RubyEncoder or loaders.

5.2 Error messages during encoding

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

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

internal encoder error,
unknown error

entire project directory when it also may contain non-Ruby files.

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

5.3 Error messages when running protected scripts

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

Ruby script '...' is protected by RubyEncoder and requires the RubyEncoder loader. Please visit the <http://www.rubyencoder.com/loaders/> RubyEncoder site to download the required loader and unpack it into '.../rgloader/' directory to run this protected script. (RuntimeError)

RubyEncoder Loaders are not installed or have been installed in the wrong directory or the script does not have enough permissions to access the rgloader/ directory or files in it. To run protected scripts on the target machine you need to install the RubyEncoder Loaders to the rgloader/ directory within the protected script directory. If your project consists of multiple directories with protected scripts in them you may install the rgloader/ directory into the project's root - protected scripts will be able to find the loaders there.

**The RubyEncoder loader is not installed. Please visit the <http://www.rubyencoder.com/loaders/> RubyEncoder site to download the required loader for '...your OS name...' and unpack it into '.../rgloader' directory to run this protected script.
in `require': no such file to load -- ... (LoadError)**

The protected script was able to locate the rgloader/ directory and run the loader.rb helper script from it, however the loader required for your platform was not found. Please [check](#) if your platform is supported and install an appropriate loader file to the rgloader/ directory.

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

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

RubyEncoder Loader - This script is not licensed to run on this machine or it is running out of

web server environment. Run this script in web server environment. Please contact the script author about this problem. Error code [02]

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

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

The protected script was encoded with a --mac locking option and was bound to some LAN hardware MAC address(es). When the script is about run on a machine running a different MAC address the above error message appears - MAC address(es) specified for the script during encoding do not match any of MAC addresses of the target machine. Check the MAC address(es) on the machine running the script. We suggest to use "ifconfig -a" command for it. Execute it in from the command line on the target machine to list all the available network interfaces. See [script locking options section](#) in this manual for details.

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

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

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

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

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

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

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

RubyEncoder supports encoding for multiple versions of Ruby. You need to use the --ruby option for this in the rubyencoder command. The above message says that the script was not encoded for the version of Ruby currently running the script. This may happen, for example if you encode the script for Ruby 1.8 (--ruby 1.8 option) and try to run it under Ruby 1.9. Make sure the script is encoded for the version of Ruby on your target server and that it is compatible with that version of Ruby (otherwise you will get an

error message during encoding). Please note, if you do not specify --ruby option in rubyencoder command, your scripts will be encoded only for Ruby 1.8.x (which is the latest stable version at the moment of writing this manual).

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

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

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

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

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

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

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

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

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

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

new license file with the same Project ID and Project Key values that were used for encoding the script.

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

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

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

The RubyEncoder Loader you installed on a target machine is expired. This may happen only with the scripts encoded by the demo version of RubyEncoder.
Download and install updated loaders from <http://www.rubyencoder.com/loaders/>

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

Your protected script was encoded with a newer version of RubyEncoder than the installed loader.
Download and install updated loaders from <http://www.rubyencoder.com/loaders/>

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

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

RubyEncoder Loader - Insufficient memory. Error code [FF]

This may happen if the RubyEncoder Loader failed to allocate some memory (RAM) using standard library functions. Check your system has enough free memory for running the protected script. RubyEncoder Loader does not need much free memory available, so this error usually never appears. Otherwise your system cannot work anyway without free memory available.

RubyEncoder Loader - Internal error: ...

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

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

RubyEncoder 3 **for Windows, MacOS and Linux**

Part



VI

6 Advanced users

6.1 Ruby shell scripts encoding

RubyEncoder keeps the first line of the script unchanged if it begins with a `#!` UNIX shell script prefix (e.g. `#!/bin/ruby`). This lets protected scripts run from the shell or as CGI scripts. The first line of the script will not be encoded but the whole script including this line will still be protected with a checksum and so remains protected from unauthorized modifications.

6.2 Marking a file to be skipped during encoding

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

```
RubyEncoder:DO_NOT_ENCODE
```

```
E.g. # RubyEncoder:DO_NOT_ENCODE
```

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

RubyEncoder 3 **for Windows, MacOS and Linux**

Part



VII

7 License

7.1 RubyEncoder License

PLEASE READ THIS CAREFULLY BEFORE USING MATERIALS

A IMPORTANT PROVISIONS

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

B PROPERTY OF SOURCEGUARDIAN

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

C LICENCE ACCEPTANCE PROCEDURE

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

D LICENCE REJECTION PROCEDURE

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

E OTHER AGREEMENTS

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

LICENCE AGREEMENT AND LIMITED WARRANTY

1. Ownership of materials and copies

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

2. Licence

2.1. Evaluation Licence

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

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

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

2.2. Full Licence

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

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

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

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

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

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

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

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

3. Limited Support

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

4. Licence restrictions

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

5. No transfer

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

6. Undertakings

You undertake to:

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

7. Limited warranty

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

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

8. No other warranties

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

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

9. Limitation of liability

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

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

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

10. Exclusion of liability

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

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

11. Your statutory rights

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

12. Term

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

13. Export

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

14. General

14.1. You agree that SourceGuardian shall have the right, after supplying undertakings as to confidentiality, to audit any computer system on which the Materials are installed in order to verify compliance with this licence Agreement.

14.2. This Licence Agreement constitutes the complete and exclusive statement of the Agreement between SourceGuardian and you with respect to the subject matter of this Licence Agreement and

supersedes all proposals, representations, understandings and prior agreements, whether oral or written, and all other communications between us relating to that subject matter.

14.3. Any clause in this Licence Agreement that is found to be invalid or unenforceable shall be deemed deleted and the remainder of this Licence Agreement shall not be affected by that deletion.

14.4. Failure or neglect by either party to exercise any of its rights or remedies under this Licence Agreement will not be construed as a waiver of that party's rights nor in any way affect the validity of the whole or part of this Licence Agreement nor prejudice that party's right to take subsequent action.

14.5. This Licence Agreement is personal to you and you may not assign, transfer, sub-contract or otherwise part with this Licence Agreement or any right or obligation under it without the SourceGuardian's prior written consent.

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

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

7.2 RubyEncoder Loaders License

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

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

1. Definitions

Defect

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

End User(s)

Users of the Loader in commercial operation

Loader

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

Maintainence

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

RubyEncoder

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

Support

Assisting the End User with enquiries relating to Defects

2. License for End Users

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

2.1 Reading scripts encrypted with RubyEncoder;

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

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

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

3. Restrictions and Exceptions

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

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

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

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

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

4. General

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

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

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

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

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

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

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

RubyEncoder 3 **for Windows, MacOS and Linux**

Part



8 Changelog

8.1 Version 3 / February 2023

Version 3 introduces support for Ruby 3.0, 3.1 and 3.2. As usual we are looking forward to hearing from our customers about suggestions for improving RubyEncoder and we are open to new ideas.

8.2 Version 2.7.5 / February 2022

Version 2.7.5 introduces new GUI and other improvements. We did not add support for new versions of Ruby in this version. As usual we are looking forward to hearing from our customers about suggestions for improving RubyEncoder and we are open to new ideas. Here is a list of the recent version 2.7.5 changes.

- A fully rebuild GUI. You will not notice many visual changes but the updated version supports hi DPI monitors and has some other minor refinements.
- Communication with rubyencoder.com website uses SSL connection now.

8.3 Version 2.7 / May 2020

Version 2.7 introduces encoding for Ruby 2.7, new loaders, new starter code and loader and bug fixes. As usual we are looking forward to hearing from our customers about suggestions for improving RubyEncoder and we are open to new ideas. Here is a list of the recent version 2.7 changes.

NEW FEATURES

- **Encoding for Ruby 2.7**

We have added full support of encoding for Ruby 2.7 including all the newest language features. In order to encode for Ruby 2.7, please select an appropriate checkbox in your project settings or use the --ruby 2.7 option if you prefer to use the command line encoder. As usual you may combine encoding for different versions of Ruby but your code must be compatible with all the selected versions.

- **We updated a built-in starter code** which you can see at the beginning of every RubyEncoder encoded file. The starter code is looking for an appropriate binary loader when the encoded Ruby code runs. Now it also checks a version number of the loader required for running the current encoded code and the loader which may be already loaded by another encoded file probably encoded with an older version of RubyEncoder. In the previous versions of RubyEncoder this may cause a problem when the newer encoder file can't run because of decoding by an already loaded older loader. Now the new starter code can re-load the newer version of the loader which is required by the current code. Newer loaders are always compatible with older encoded files and can run both old and new encoded files.
- Minor update for directories recursive search on Windows for the CLI version. Directories might appear in the log with an error which did not cause any issues though, we have fixed it in the new version.

- For the CLI version `--ruby X.Y` option was updated when used for excluding a range of target Ruby versions. This works as follows:
 - `--ruby x.y` adds x.y version only (no changes)
 - `--ruby x.y+` (plus sign) adds x.y and higher versions (no changes)
 - `--ruby x.y-` (minus sign) skips x.y version only (new)**
 - `--ruby x.y--` (double minus sign) skips x.y and lower versions (previously worked with a single minus)**

Note, the order `--ruby` options appear in the command line does matter.

- `RGLoader::rg_get_const()` API function returns an array of all the constants if no arguments are given
- **MacOS Catalina support**
- Optimized support for Ruby 2.6 and 2.7 global variables.

SUPPORTED RUBY VERSIONS

- **Encoding for Ruby 1.8.7 to 2.7 are fully supported**

SUPPORTED OS

- Encoder is available for MacOS including MacOS 10.15 Catalina, Linux and Windows.
- We removed support for i386 32-bit MacOS
- We reverted support for Linux i386 32-bit platform since you asked us for it.
- GUI and command line encoders and tools are included.
- Loaders are available for desktop and server platforms running MacOS, Linux, FreeBSD, Windows (rubyinstaller.org MinGW), Windows (native) and embedded platforms running ARMel Linux, ARMhf Linux (Raspberry Pi, BeagleBoard etc)

8.4 Version 2.6 / November 2019

Version 2.6 introduces encoding for Ruby 2.6, new loaders and bug fixes. As usual we are looking forward to hearing from our customers about suggestions for improving RubyEncoder and we are open to new ideas. Here is a list of the recent version 2.6 changes.

NEW FEATURES

- Encoding for Ruby 2.6

We have added full support of encoding for Ruby 2.6 including all the newest language features. In order to encode for Ruby 2.6, please select an appropriate checkbox in your project settings or use the `--ruby 2.6` option if you prefer to use the command line encoder. As usual you may combine encoding for different versions of Ruby but your code must be compatible with all the selected versions.

- MacOS Catalina support

- Optimized support for arrays and hashes which fixes issues with encoding of some Ruby source files containing long arrays. Also the encoded files containing long arrays can be loaded faster now.
- As of version 2.5 of RubyEncoder UTF-8 encoding is used by default if nothing else is specified in the --encoding option or the same option in GUI in Advanced settings. Since version 2.6 UTF-8 is used as a default encoding for the code itself or the encoding specified in the --encoding option is used for the code as well. You can use non-ASCII names for constants and since Ruby 2.6 non-ASCII names are supported for classes and methods.

SUPPORTED RUBY VERSIONS

- Encoding for Ruby 1.8.7 to 2.6 are fully supported

SUPPORTED OS

- Encoder is available for MacOS including MacOS 10.15 Catalina, Linux (x86_64) and Windows.
- We removed support for i386 32-bit MacOS and Linux platforms.
- GUI and command line encoders and tools are included.
- Loaders are available for desktop and server platforms running MacOS, Linux, FreeBSD, Windows (rubyinstaller.org MinGW), Windows (native) and embedded platforms running ARMel Linux, ARMhf Linux (Raspberry Pi, BeagleBoard etc)

8.5 Version 2.5 / January 2019

Version 2.5 introduces encoding for Ruby 2.5, new loaders, a lot of new command line and GUI features. As usual we are looking forward to hearing from our customers about suggestions for improving RubyEncoder and we are open to new ideas. Here is a list of the recent version 2.5 changes.

NEW FEATURES

- Encoding for Ruby 2.5

We have added full support of encoding for Ruby 2.5 including all the newest language features. In order to encode for Ruby 2.5, please select an appropriate checkbox in your project settings or use the --ruby 2.5 option if you prefer to use the command line encoder. As usual you may combine encoding for different versions of Ruby but your code must be compatible with all the selected versions.

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

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

Command line encoder updates

- We added automatic registration for command line tools. So, if you install the no-GUI package (Linux) or get an additional license for using command line tools on another machine, server etc e.g. to generate custom licenses there, you may use automatic registration on the first run. Run rubyencoder command line executable or licgen tool, read and agree with the terms, enter your RubyEncoder online user account email and password when asked in the terminal. If automatic registration can't be used for any reason, e.g. if there is no Internet access, register software as usual. Copy the hex code from the screen, paste it to the user profile, download the encode.lic license file and copy it to the binary folder where the executable is located.
- The command line tools now may be started with GUI license. It's not necessary to specify a path to the GUI encode.lic license file anymore using the -L option. The GUI license will be automatically found if you are starting the command line tools included to GUI installation. However, if you are installing the command line tools separately or to another machine, please register your copy as usual and install the encode.lic license file to the same binary folder where the tools are located or use automatic registration for new copies.
- The license may be released automatically from for the command line tools as well. Please read below as we added the same option to the GUI version. In order to release the license from the command line, use --license-release option with rubyencoder command line tool or licgen and follow the instructions displayed on your terminal.
- We added a new -c (--copy) filter option in addition to -f (--file) and -x (--exclude). The new -c (--copy) option may be used to specify the files that will be copied as-is without encoding to the target folder. This option makes sense only if you specify the target folder with -o (--output) option. If -c is used without -o, then it works as -x and skips the specified files. The new option may take * and ? wildcards or a @filelist.

E.g. you may now encode *.rb files but keep some of them unencoded and copy, e.g. configuration files:

```
/path/to/rubyencoder -o /path/to/target -f "*.rb" -c "config.rb" /path/to/source
```

- Optional directory trimming level may be specified with -r{n} The default is 0 and means no trimming, this matches the mode used in previous versions of RubyEncoder. If n is specified, the encoder will remove n folder names from the beginning of target file paths when encoding or copying the files to the target folder. This is similar to -p option of patch utility on Unix. You may find some [samples](#) useful.
- We updated recursive directory search. Source paths containing wildcards in directory names now work, e.g. /path/to/dir?/*.*rb This also works in @filelist and you may use it with -f, -c, -x. If the @filelist is specified, recursion is automatically turned on, but it's still possible to change the directory trimming level with -r{n} if necessary.
- {RG_EXPIRY_DATE} tag was added and may be used in the custom text within the generated license files. Use this tag to embed your custom license expiry date into the readable part of the license. Changing of this text in the license file does not make sense and is NOT opening a back door, if you use the expiry date in your custom licenses. The expiry date is stored encrypted as well as other options within the license file. The readable text is only for information and you may put a name of your product, name of your customer and now the expiry date. For further details please see [Advanced Options](#) in GUI or --text option for the [command line licgen tool](#).

- We updated how empty files are processed. Now empty files are copied to the target folder if it's specified. Empty files are skipped from processing if the target folder is not specified. In either case empty files are not counted as errors anymore. The encoding log will still indicate empty files with [11] code.
- We added the `--verbose` option to the command line encoder. Options are 0-quiet, 1-print only errors in the log, 2-print standard log. 2 is default. You may also add this option to "Additional Command Line Options" in "Advanced settings" if you want to change the encoding log when using GUI.
- A minor addition which may be a great help for the users who encode from the command line. We added "+" and "-" options for the `--ruby`. "+" means to encode for the specified version of Ruby and for all the newer versions which are supported by the current version of RubyEncoder. "-" means to encode for all the supported versions of Ruby except the specified one and all the lower versions, which is useful if you always need to encode by default for new versions but do not need support for old versions starting from some one. E.g.

```
--ruby 2.4.0+  encodes for Ruby 2.4 and all the newer versions (up to 2.5 for this version)
--ruby 2.0.0+  encodes for Ruby 2.0 to 2.5 and newer (up to 2.5 for this version)
--ruby 2.3.0-  encodes for Ruby 2.4 and newer, i.e. excludes Ruby 2.3 and older
```

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

- Instead of using the `--days` option in the command line you may now use a more precise `--expire 00d00m00h00s`
- UTF-8 encoding is now used by default if nothing else is specified.
- We are introducing a new locking option which is available in the full version of the encoder - locking to a machine ID. Use the new loader method `RG_Loader::get_machine_id()` to obtain a machine ID on your client machine and then specify the machine ID on the Lock screen in GUI or use the new command line option `--machine-id` of the encoder or `licgen` tool. Encoded Ruby scripts locked to a machine ID will only run on that machine (or machines if you specify multiple machine IDs). Please refer to the [Locking options](#) section for further information.

Note: as `RG_Loader::get_machine_id()` is a binary method of the loader, an appropriate loader must be installed to the client machine in order for this method to be available from your code. We recommend that you create a mini project, encode it and include the loaders for obtaining the machine ID from the client, in that case loaders will be found automatically as for any other Ruby script encoded with RubyEncoder.

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

For this to work you need to do two things:

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

```
print RGLoader::get_verification_id()
```

Note: if you use any frameworks, you may need to use another mechanism for sending a string to the output instead of print, e.g. *res.write* for Cuba etc

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

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

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

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

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

- If you are using locking to MAC addresses, the new `RGLoader::get_mac_addresses()` method may be used on the client machine to obtain MAC address and then you use them in the locking settings or automatic license generation with `licgen` tool.

Note: as `RG_Loader::get_mac_addresses()` is a binary method of the loader, an appropriate loader must be installed to the client machine in order for this method to be available from your code. We recommend that you create a mini project, encode it and include the loaders for obtaining the MAC addresses from the client's machine for further using them for locking your code. In that case loaders will be found automatically as for any other Ruby script encoded with RubyEncoder.

GUI updates

- We fully reworked the files and folders selection dialog which is used when you add files or folder to the project. This new dialog also uses predefined file filters which are based on your filter settings in File/Preferences.
- We fully reworked files and folders highlighting in the project tree. Folders - bold, virtual folders - green, files/folders changed or added since last encoding - blue.
- Newly added folders and files will be encoded at least once if the "encode only modified" option is selected in advanced settings.
- The option to encode only changed files (in Advanced Settings) always checks modification date for all the files in the project before encoding.
- The project is automatically checked for new and deleted files every time you open the project. This may be turned off/on in [Preferences](#).

- We added samples for code sections in [Advanced Options](#) - code for loader not installed, custom Ruby header and error handler and custom license text. Try the "Want sample" buttons after clicking "Edit" for the code you want to change.
- We added a new "Copy unencoded" filter section to [Preferences](#). These filters are checked before any further processing but after the exclusion filters. So, files that match any of "copy unencoded" file masks will be copied to the target folder as-is without encoding. E.g. you may now easily encode *.rb files but keep *.erb or config.rb unencoded without manually selecting encoding mode in the project tree.
- If you are locking to a license file, now it's possible to select the folder where the license will be created when you click "Generate License" on the "Lock" screen. Also RubyEncoder will remind you to generate a license file after encoding, this option may be turned off/on in Preferences.
- We added tooltips to the main project window, advanced settings and some features on the lock screen to help our new users.
- You may automatically release the current RubyEncoder license directly from the application. Please find the "Release License" button added to Help/Registration Information. You will be asked for confirmation. Releasing the license lets you reinstall the encoder to another machine or to the same machine after upgrading hardware or OS. If you are going to upgrade the machine or OS, please firstly release the license and then you may transparently re-install your copy of RubyEncoder when the upgrade is complete.

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

- File associations work now on Windows and Mac OS. It means clicking a .rg project file in Explorer/ Finder will launch RubyEncoder and opens this project in GUI. Note, for this to work, RubyEncoder must be installed using the installer on Windows, on Mac OS, you need to launch RubyEncoder at least once for file associations to start working.
- Hex registration code is now displayed in Help/Registration information which is useful if you are using multiple installations of RubyEncoder and need to manage or reset a particular license in the online user profile, now you may easily find it there.
- You may access your RubyEncoder online user profile directly from the application, click Help/ RubyEncoder User Profile.

FIXES

- Fixed encoding of hashes with non-ASCII symbols. Note, the encoding must be specified in Advanced settings in GUI or --encoding option for the command line encoder. UTF-8 is now used by default if nothing else is specified.
- Encoding of Unicode properties did not work, were not recognised by the encoder. Now it's fixed, requires a correct encoding setting in the encoder.
Sample: `string.match?(/p{Hebrew}/p{Arabic}/p{Cyrillic}/)`
- Named args were not working if function was defined without (). Now it's fixed.
Sample: `def foo bar:, baz:`

- RegExp named captures did not work. Now it's fixed.
Sample: `/(?<abc>.*)def$/`
- Better error handling in the encoder, a correct file name is displayed in the encoding log instead of `__FILE__:xx: syntax error` etc errors.

SUPPORTED RUBY VERSIONS

- Encoding for Ruby 1.8.6 to 2.5 are fully supported

SUPPORTED OS

- Encoder is available for MacOS, Linux (i386 and x86_64 versions) and Windows.
- GUI and command line encoders and tools are included.
- Loaders are available for desktop and server platforms running MacOS, Linux, FreeBSD, Windows (rubyinstaller.org MinGW), Windows (native) and embedded platforms running ARMel Linux, ARMhf Linux (Raspberry Pi, BeagleBoard etc)

8.6 Version 2.4 / February 2017

Version 2.4 introduces encoding for Ruby 2.4, new loaders and fixes some problems. As usual we are looking forward to hearing from our customers about suggestions for improving RubyEncoder and we are open to new ideas. Here is a list of the recent version 2.4 changes.

NEW FEATURES

- Encoding for Ruby 2.4

We have added full support of encoding for Ruby 2.4 including all the newest language features. In order to encode for Ruby 2.4, please select an appropriate checkbox in your project settings or use the `--ruby 2.4` option if you prefer to use the command line encoder. As usual you may combine encoding for different versions of Ruby but your code must be compatible with all the selected versions.

Please note, as Ruby 2.4 is not supported by RubyInstaller.org at the moment of releasing RubyEncoder 2.4, loaders for MinGW (rubyinstaller version) are not included to the installation package. Once the new version of RubyInstaller becomes available, we will build updated loaders for MinGW Ruby 2.4 and make them available for downloading from the website: <http://rubyencoder.com/loaders>

- Dependencies on GLIBC 2.6+ removed from encoders on Linux. This must help running RubyEncoder on some old Linux systems.
- Fixed issues with searching the license file if the path to the encoded Ruby script contains non-ASCII characters. Only Windows versions of the loaders were affected. Unfortunately, we were not able to fix the issue with non-ASCII characters in `RG_LIC_PATH` as that was caused by Ruby itself (<https://bugs.ruby-lang.org/issues/9715>, <https://bugs.ruby-lang.org/issues/12650>). Again, only Windows systems are affected by that problem.

- Fixed the old float values issue in the old Ruby 1.8.x 64-bit encoder on Linux. The issue was caused by the problem in Ruby code itself and particularly `ruby_strtod()` function. If you use 64-bit RubyEncoder on Linux for encoding of any Ruby 1.8.x scripts, this is a sample code to check: `'puts Math.log(0.5)'`. If your code is affected by that problem, please re-encode it with RubyEncoder 2.4.
- Fixed [encoding of only files changed since last encoding](#) in GUI. This must work now without re-opening the project.
- Loaders were updated for all supported OS and versions of Ruby.

SUPPORTED RUBY VERSIONS

- Encoding for Ruby 1.8.6 to 2.4 are fully supported

SUPPORTED OS

- Encoder is available for MacOS, Linux (i386 and x86_64 versions) and Windows.
- GUI and command line encoders and tools are included.
- Loaders are available for desktop and server platforms running MacOS, Linux, FreeBSD, Windows (rubyinstaller.org MinGW), Windows (native) and embedded platforms running ARMel Linux, ARMhf Linux (Raspberry Pi, BeagleBoard etc)

8.7 Version 2.3 / December 2016

Version 2.3 introduces encoding for Ruby 2.3, new loaders and fixes some problems. As usual we are looking forward to hearing from our customers about suggestions for improving RubyEncoder and we are open to new ideas. Here is a list of the recent version 2.3 changes.

NEW FEATURES

- Encoding for Ruby 2.3

We have added full support of encoding for Ruby 2.3 including all the newest language features. In order to encode for Ruby 2.3, please select an appropriate checkbox in your project settings or use the `--ruby 2.3` option if you prefer to use the command line encoder. As usual you may combine encoding for different versions of Ruby but your code must be compatible with all the selected versions.

- If the target Ruby version is not specified in the command line using the `--ruby` option, then encoding will be done for Ruby 2.0 to 2.3. Please specify the `--ruby 1.8` or `--ruby 1.9.x` option if you need to encode for old versions of Ruby.
- Loaders were updated for all supported OS and versions of Ruby. New loaders added for Ruby 2.3 including 32 and 64-bit MinGW loaders for Ruby installations from rubyinstaller.org. These loaders are now default for the Windows platform.

- rgloader/loader.rb starter script was slightly changed to refine detection of custom Ruby versions and two digits minor Ruby versions.
- The command line tools now return expected exit codes. The encoder returns encoding status after processing a single file. When it is processing multiple files it returns zero in case of running the process and if there are no issues in using the command line options and then you need to check the encoding log for further details. Licgen returns exit code on invalid options or status of the license generation. Rginfo returns exit code on invalid options or status of the encoded file. Please find further details in the [Exit codes](#) section.
- The dynamic loader code that is added by default to every encoded file now checks if RubyEncoder loader is already started and does not try to find and load it again. This may improve performance for nested protected files.
- You may exclude the default dynamic loader code from protected files if you are starting the loader manually from your other code e.g. if you want to start the loader from the custom folder. Please find the [new option in GUI](#) and [command line](#) for that. Please note, if you use this option you must start the loader before running the encoded file.

SUPPORTED RUBY VERSIONS

- Encoding for Ruby 1.8.6 to 2.3 are fully supported

SUPPORTED OS

- Encoder is available for MacOS, Linux (i386 and x86_64 versions) and Windows.
- GUI and command line encoders and tools are included.
- Loaders are available for desktop and server platforms running MacOS, Linux, FreeBSD, Windows (rubyinstaller.org MinGW), Windows (native) and embedded platforms running ARMel Linux, ARMhf Linux (Raspberry Pi, BeagleBoard etc)

8.8 Version 2.2 / February 2015

Version 2.2 introduces encoding for Ruby 2.1 and Ruby 2.2, new loaders and fixes some problems. As usual we are looking forward to hearing from our customers about suggestions for improving RubyEncoder and we are open to new ideas. Here is a list of the recent version 2.2 changes.

NEW FEATURES

- Encoding for Ruby 2.1 and Ruby 2.2

We have added full support of encoding for Ruby 2.1 and 2.2 including all the newest language features. In order to encode for Ruby 2.1, please select an appropriate checkbox in your project settings or use the --ruby 2.1 option if you prefer to use the command line encoder. Select the Ruby 2.2 checkbox or use the --ruby 2.2 option accordingly in order to encode for Ruby 2.2. As usual you may combine encoding for different versions of Ruby but your code must be compatible with all the selected versions.

Earlier we released a version of loaders that partially supported Ruby 2.1. Now we added full support. In order to use new language features, please re-encode your files and install the latest loaders.

- If the target Ruby version is not specified in the command line using the `--ruby` option, then encoding will be done for Ruby 2.0 to 2.2. This differs from the previous version and we do not encode for Ruby 1.8.x or 1.9.x by default anymore. Please specify the `--ruby 1.8` or `--ruby 1.9.x` option if you need to encode for old versions of Ruby.
- Loaders were updated for all supported OS and versions of Ruby. New loaders added for Ruby 2.1 and 2.2 including 32 and 64-bit MinGW loaders for Ruby installations from rubyinstaller.org. These loaders are now default for the Windows platform. If you use an old native windows build of Ruby or run your custom build compiled with native windows Visual C compiler and VC libs, please download loaders from our website.
- `rgloader/loader.rb` starter script was slightly changed to refine detection of custom Ruby versions.

SUPPORTED RUBY VERSIONS

- Encoding for Ruby 1.8.6 to 2.2 are fully supported

SUPPORTED OS

- Encoder is available for MacOS, Linux (i386 and x86_64 versions), FreeBSD (i386 and x86_64 versions) and Windows.
- GUI and command line encoders and tools are included.
- RubyEncoder for FreeBSD is available as command line tools.
- Loaders are available for desktop and server platforms running MacOS, Linux, FreeBSD, Windows (rubyinstaller.org MinGW), Windows (native) and embedded platforms running ARMel Linux, ARMhf Linux (Raspberry Pi, BeagleBoard etc)

8.9 Version 2.0 / September 2013

Version 2.0 introduces encoding for Ruby 2.0, new loaders and fixes some problems. As usual we are looking forward to hearing from our customers about suggestions for improving RubyEncoder and we are open to new ideas. Here is a list of recent version 2.0 changes.

NEW FEATURES

- Encoding for Ruby 2.0

We have added encoding for Ruby 2.0 including all the newest language features. In order to encode for Ruby 2.0, please select an appropriate tickbox in your project settings or use the `--ruby 2.0` option if you prefer to use the command line encoder.

- If the target Ruby version is not specified in the command line using the `--ruby` option, then encoding will be done for Ruby 1.9.x and 2.0. This differs from the previous version and we do not encode for

Ruby 1.8.x by default anymore. Please specify the `--ruby 1.8` option if you need to encode for this version of Ruby.

- Loaders were updated for all supported OS and versions of Ruby. New loaders added for Ruby 2.0 including 32 and 64-bit MinGW loaders for Ruby installations from rubyinstaller.org
- Minor changes of how the command line encoder recursively walk the directory tree. Depending on availability in OS `getcwd()` is used before checking the `PWD` environment variable for getting the current directory. This fixes issues with the directory recursion not working in some specific custom environments.
- Standard input and output support for the [command line encoder](#).
- Added "Script will expire in (days)" option to the [Locking page](#).
- Changed how 'modification date' works in GUI
- The 'Copy unencoded' mode is now assigned to empty files when adding them to the project. This is to eliminate 'empty file - skipped' error messages from the encoder. It replicates what GUI does with other files that are not detected by their file extensions. If you don't need to copy empty files to the target folder, you may change the encoding mode to 'Skip' for them.
- Reverting the project was affecting modification dates - fixed
- Updated built-in support and automatic update in GUI.

SUPPORTED RUBY VERSIONS

- Encoding for Ruby 1.8.6 to 2.0 are fully supported

SUPPORTED OS

- Encoder is available for MacOS, Linux (i386 and x86_64 versions), FreeBSD (i386 and x86_64 versions) and Windows.
- GUI and command line encoders and tools are included.
- RubyEncoder for FreeBSD is available as command line tools.
- Loaders are available for desktop and server platforms running MacOS, Linux, FreeBSD, Windows (native), Windows MinGW and embedded platforms running ARMel Linux, ARMhf Linux (Raspberry Pi, BeagleBoard etc)

8.10 Version 1.5 / March 2013

Version 1.5 introduces GUI for RubyEncoder for MacOS, Linux and Windows, fixes problems and adds some new options. The update is partly based on comments and suggestions of our users. We were glad to receive comments and suggestions and want to thank you very much for sharing your ideas. We are looking forward to hearing about other suggestions for improving RubyEncoder and we are open to new ideas. Here is a list of recent version 1.5 changes.

NEW FEATURES

- Specifying a target character encoding

Target encoding may be specified during encoding in Advanced options in GUI or using the new --encoding option in the command line. RubyEncoder compiles source Ruby files to a binary representation, as a result 'magic comments' cannot be used to specify character encoding of source files. Please [use this new RubyEncoder option](#) when you encode files that have 'magic comments' for specifying the character encoding.

- Rails compatibility option

A new Rails compatibility option was [added](#). Enabling Rails compatibility lets you encode all Rails *.rb files. Normally you can encode only application controllers, model and helper files. Other files if encoded would not work under Rails if the new Rails compatibility mode was not used. Only Ruby files can be encoded anyway. The option may be enabled in Advanced options in GUI or using the --rails command line option.

- Option to stop encoding at a critical error

We have added a new option for the encoder to stop at a critical error. [See details](#).

- A new option to encode only changed files detected by file modification date. [See details](#).

- Automatic filtering UTF-8 BOM from all source files. This allows encoding files created in editors that save BOM.

- Marking a file to be skipped during encoding

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

```
# RubyEncoder:DO_NOT_ENCODE
```

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

- Now it is possible to send licgen's output to console instead of a file, use -- (double dash) instead of the output file name. [See details](#)

- Added a new license generator option to add custom text into the license file.

We have added an option to include custom text into license files generated by RubyEncoder license generator. The included text is protected with a checksum against modification. The option may be used to include user information, license description etc into license files.

- Custom constants substitution in the custom header code. All user {constants} will be replaced in the prepend code. Also some standard RG constants may be used: {RG_DATE} - current date i.e. date of encoding, {RG_LICENSEE} - RubyEncoder license owner from the RubyEncoder license file. It works in the same way also for licgen and do replacements for custom text if it is used.

GUI FOR MacOS, LINUX AND WINDOWS

- We are proud to present a new cross platform GUI for MacOS, Windows and Linux versions of RubyEncoder. It includes all the features of the command line encoder and tools and even more. It has built-in support system and more.

SUPPORTED RUBY VERSIONS

- Encoding for Ruby 1.8.6 to 1.9.3 are fully supported

SUPPORTED OS

- Encoder is available for MacOS, Linux (i386 and x86_64 versions), FreeBSD (i386 and x86_64 versions) and Windows.
- GUI and command line encoders and tools are included.
- RubyEncoder for FreeBSD is available as command line tools.

Index

- A -

About RubyEncoder™ 2

- B -

Backup 64
Binding 53
Buy 2

- C -

CGI 85
Command line encoder installation 43
Command line encoder installation for FreeBSD 45
Command line encoder installation for Linux and FreeBSD 44
Command line encoder installation for Mac OSX 43
Command line encoder installation for Windows 43

- E -

encode.lic 48
Encoded scripts modification 78
Encoder errors 78
Error 78
Error messages during encoding 78
Error messages during protected scripts run 79
Errors, common mistakes and possible solutions 78
Evaluation scripts 53
Exclude files 64
External license file 53, 69

- F -

Features 2
First run 48
FreeBSD 44, 45

- G -

GUI manual overview 7

- H -

How it works 2
How to buy 2

- I -

Install 75
Installation 43, 44, 45
Installing loaders 75

- L -

License 48, 64
licgen 69, 70
Linux 44
Loader errors 78, 79
Loaders 75
Locking 2, 53
Locking to domain 53
Locking to IP 53
Locking to MAC address 53

- M -

Mac 43

- O -

Obtaining license 7
Options 2
OSX 43
Other options 64
Output directory 64

- P -

Project ID 53
Project Key 53
Protected script loaders 75

Purchase 2

- R -

rginfo 72

rgloader directory structure 75

Ruby shell scripts encoding 85

rubyencoder 48, 49, 53, 64

Running the command line encoder 48

- S -

Script locking options 53

Setup 43, 44, 45, 75

Startup screen 9

Supported OS and platforms 75

- T -

Time server 53

- U -

Usage of Information Tool 72

Usage of License Generator 70

Usage of RubyEncoder 49

Using external script license generator 69

Using information tool 72

- V -

Version 64

- W -

Windows 43