8 Changelog

8.1 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 additon 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 Mac OS X, 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 Mac OS X, Linux, FreeBSD, Windows (rubyinstaller.org MinGW), Windows (native) and embedded platforms running ARMel Linux, ARMhf Linux (Raspberry Pi, BeagleBoard etc)