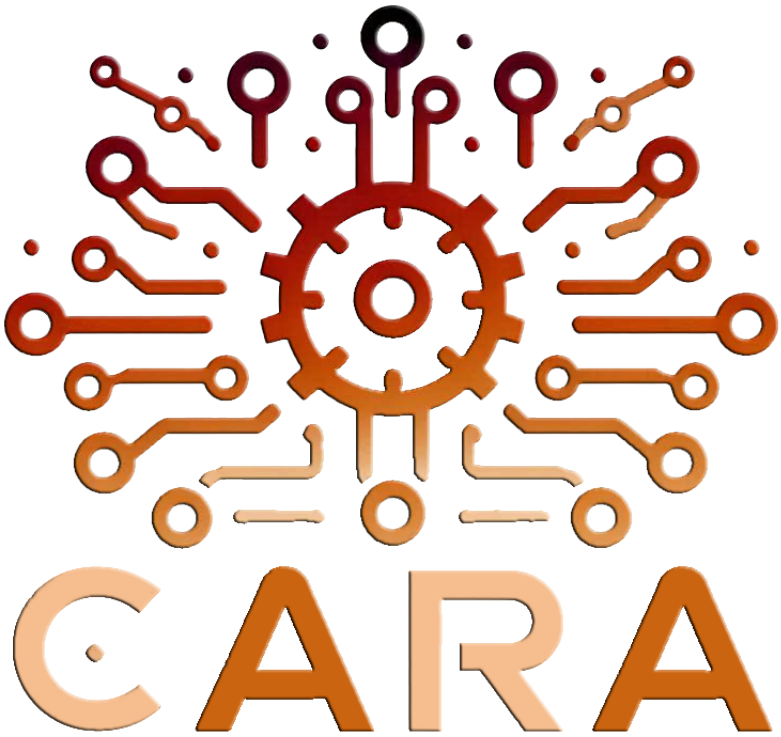

Changelog Automation & Release Assistant (CARA)
Manual & Programming Documentation

Developer: Antonius Torode

Version – Version 0.001
Latest update: July 1, 2025

The Original Maintainer of this work is: Antonius Torode.

The Current Maintainer of this work is: Antonius Torode.

This document is intended solely to provide documentation for the Changelog Automation & Release Assistant (CARA), a tool developed to automate changelog generation and formatting using Git history. Throughout this document, "CARA" will refer to the Changelog Automation & Release Assistant. This tool was created for configurable changelog output and supports multiple logging and formatting schemes for software projects.

This document is continuously under development.
Most Current Revision Date: July 1, 2025

Torode, A.
CARA Documentation.
2025.
Includes Source Code and References

Contents

This page intentionally left blank.
(Yes, this is a contradiction)

Chapter 1

Introduction

The **Changelog Automation & Release Assistant (CARA)** is a command-line tool designed to automate the process of generating and maintaining changelogs using Git commit history. CARA parses Git logs and formats the output according to a customizable configuration, allowing users to streamline their release documentation process.

CARA supports multiple verbosity levels, debug output, and configurable input/output paths, making it adaptable to a variety of development workflows. It is intended for developers who maintain changelogs regularly and prefer consistency, automation, and clarity in their version history tracking.

1.1 How to Use CARA

To use **CARA**, ensure the main script `cara.py` and a configuration file (e.g., `cara.conf`) are present in your project directory. The configuration file specifies options such as grouping method, output format, filtering rules, and more. See chapter 2 for more details on the configuration file. Once configured, run the script from your project's root directory using:

```
python3 /path/to/cara.py -c /path/to/cara.conf
```

CARA will parse your Git commit history, group entries according to your configuration (e.g., by day or week), filter out undesired commits, and generate a structured changelog. The output will be written to the file path specified in your command. For full details on configuration options, refer to the documentation in the `docs/` directory. Various other optional program arguments are available which can be accessed via the `-h` flag:

```
$ python3 cara.py -h
usage: cara.py [-h] [-v] [-d] [-c CONFIG] [-i INPUT] [-o OUTPUT] [-r
    ↪ REPO]

Changelog Automation & Release Assistant (CARA).

optional arguments:
  -h, --help            show this help message and exit
  -v, --verbose         Enables verbose mode. This will output various
     ↪ program data for
                        detailed output.
  -d, --debug           Enables debug mode. This will output various
     ↪ program data for
                        debugging.
  -c CONFIG, --config CONFIG
                        The configuration file to use.
  -i INPUT, --input INPUT
                        The input changelog to use. Use this option to
                           ↪ overwrite/update an
                        existing changelog.
  -o OUTPUT, --output OUTPUT
```

```
                        The output file to use. Use this option to
                           ↪ create a new changelog.
  -r REPO, --repo REPO  The repo path to use. This will default to the
     ↪ current directory.
```

Chapter 2

Configuration

The CARA configuration system allows users to define how changelogs are generated and formatted. Configuration files provide a flexible and human-readable way to customize the behavior of the application without modifying the source code.

CARA reads configuration data from a plain text file, where each line defines a key-value pair separated by an equals sign (‘=‘). Lines beginning with ‘#‘ are treated as comments and ignored. Empty lines and improperly formatted entries are also skipped during parsing.

The configuration system is implemented through the `Config` class, which is responsible for loading, parsing, and storing configuration values. Users can specify a configuration file at runtime via the `-config` flag. Once loaded, the configuration can be queried programmatically or used internally to guide CARA's behavior. To use the default behavior for any configuration option, simply omit or comment out the option from the configuration file.

The expected format for configuration entries is:

```
key=value  # Optional comment.
```

This format provides clarity and simplicity, ensuring easy editing and version control. Each configuration key corresponds to a specific setting or feature within CARA, as documented in subsequent sections of this chapter.

2.1 DATE_FORMAT Configuration Option

The `DATE_FORMAT` option in the configuration file allows users to control the format in which commit dates are displayed in the generated changelog. This option directly maps to the `-date=format:<str>` flag in `git log`, enabling a wide range of formatting options based on user preferences or documentation standards. If this option is defined in the configuration file, CARA will invoke `git log` with the specified format string. Otherwise, Git's default date formatting is used.

```
// Example Configuration
DATE_FORMAT=%Y-%m-%d
```

Common Format Examples

- `%Y-%m-%d` – 2025-07-01 (ISO standard)
- `%d-%m-%Y` – 01-07-2025 (European)
- `%B %d, %Y` – July 01, 2025 (Verbose)
- `%a %Y-%m-%d at %H:%M` – Tue 2025-07-01 at 14:30

For the full list of supported format tokens, refer to the `strftime(3)`[1] man page or Git's documentation on custom date formats.

2.2 GROUP_BY Configuration Option

The `GROUP_BY` configuration value determines how commit entries in the generated changelog are grouped. Grouping provides structure to the changelog output by organizing commits into logical sections based on time.

Accepted Values

- `day` — Groups commits by individual date (e.g., `2025-07-01`).
- `week` — Groups commits by ISO calendar week (e.g., `2025-W27`).
- `month` — Groups commits by calendar month (e.g., `2025-07`).
- `year` — Groups commits by year (e.g., `2025`).

Default Behavior

If this configuration value is not set, the default grouping is by `day`.

Usage

This option affects the organization of the changelog. Each group is rendered as a section in the output, prefixed with a heading containing the grouping key (e.g., date or week label). Commits within each group are listed chronologically.

2.3 OUTPUT_ENTRIES Configuration Option

The `OUTPUT_ENTRIES` configuration option controls which fields are included for each commit entry in the generated changelog output. This value should be a space-separated list of field names, allowing flexible control over the formatting of each entry.

Valid Fields:

- `commit` – Includes the full commit hash.
- `author` – Includes the name of the author who made the commit.
- `message` – Includes the commit message.

Special Value:

- `all` – A shortcut that includes all of the above fields in the default format.

Example Values:

- `OUTPUT_ENTRIES=commit message`
- `OUTPUT_ENTRIES=author`
- `OUTPUT_ENTRIES=all`

Each selected field will be printed in order, separated by colons and dashes according to default formatting. This allows for compact or verbose output depending on user preference.

2.3.1 Filtering Configuration Options

The changelog generator supports several filtering options to control which commit messages are included in the final output. These options allow users to exclude trivial or irrelevant commits and focus on meaningful changes.

MIN_WORDS Specifies the minimum number of words required in a commit message for it to be included. If the message contains fewer words, it will be discarded. This option takes priority over `MIN_CHARS` if both are set.

MIN_CHARS Specifies the minimum number of characters required in a commit message. If the message is shorter than this length, it will be excluded. Ignored if `MIN_WORDS` is also set.

EXCLUDE_KEYWORDS A space-separated list of keywords. If any of these appear in a commit message (case-insensitive), the message will be excluded from the output.

INCLUDE_KEYWORDS A space-separated list of required keywords. Only commit messages containing at least one of these keywords (case-insensitive) will be included. If not set, this filter is ignored.

These options provide a flexible way to tailor the changelog content by filtering out uninformative or irrelevant commits.

References

[1] https://www.man7.org/linux/man-pages/man3/strftime.3.html