



Version – Version 0.001  
Latest update: July 1, 2025

© 2025 Antonius Torode  
All rights reserved.

This work may be distributed and/or modified under the conditions of Antonius' General Purpose License (AGPL).

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

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Configuration</b>	<b>2</b>
2.0.1	DATE_FORMAT Configuration Option . . . . .	2

*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.

# 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:

```
1 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.

### 2.0.1 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.

```
1 // Example Configuration
2 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)` man page or Git's documentation on custom date formats.

# References

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