

RustLogs (RLG) is a powerful and flexible logging library for Rust that provides application-level logging with a simple and readable output format. It offers a wide range of features and customization options to streamline your logging workflow and enhance your Rust applications.

Overview

RustLogs (RLG) simplifies the process of adding logging functionality to your Rust applications. With its intuitive APIs and helpful macros, you can easily integrate logging into your codebase and gain valuable insights into your application's behavior.

divider

Features

Multiple Log Levels RustLogs (RLG) supports a comprehensive set of log levels, including

,
,
,
,
,
,
,
,
,
, and

, allowing you to control the granularity of your logs.

Structured Log Formats The library provides structured log formats that are easy to parse and filter, enabling efficient analysis and troubleshooting of your app

lication's logs.

Extensive Output Format Compatibility Rust Logs (RLG) supports a wide range of output formats, including

Common Event Format (CEF) - A standardized log format developed by ArcSight for the exchange of event information between security devices and applications. Extended Log Format (ELF) - A log format that extends the Common Log Format (CLF) with additional fields to provide more detailed information about web server requests. Graylog Extended Log Format (GELF) - A log format designed for use with the Graylog log management system, which allows for the transmission of log messages over a network. JavaScript Object Notation (JSON) - A lightweight data interchange format that is easy for humans to read and write and easy for machines to parse and generate. NCSA Common Log Format (CLF) - A standardized log format used by web servers to log client requests, including information such as the client IP address, request method, and response status code. W3C Extended Log File Format (W3C) - A log format developed by the World Wide Web Consortium (W3C) that extends the Common Log Format (CLF) with additional fields for more detailed logging of web server requests. Syslog Format - A standardized log format used by various network devices and applications to send event messages to a centralized logging server. Apache Access Log Format - A log format used by the Apache web server to log client requests, including information such as the client IP address, request method, and response status code. Logstash Format - A log format used by the Logstash data processing pipeline, which allows for the collection, parsing, and forwarding of log data from various sources. Log4j XML Format - A log format used by the Log4j logging framework, which allows for the structured logging of events in an XML format. NDJSON (Newline Delimited JSON) - A log format where each log

message is represented as a JSON object on a separate line, making it easy to parse and process log data in a streaming fashion.

Flexible Configuration

Customize the behavior of RustLogs (RLG) to suit your application's needs. You can easily configure the log file path, log levels, and output formats to match your requirements.

Asynchronous Logging

RustLogs (RLG) supports asynchronous logging, allowing you to log messages without blocking your application's execution. This ensures optimal performance and responsiveness.

Helpful Macros

The library provides a set of convenient macros that simplify common logging tasks, such as creating log entries, printing logs to stdout, logging to files, and conditionally logging based on feature flags or predicates.

divider

Get Started with RustLogs (RLG) Today

Ready to take your Rust application logging to the next level? Get started with RustLogs (RLG) today and experience the power and flexibility of advanced logging!

Easy Integration

Integrating RustLogs (RLG) into your Rust project is a breeze. With just a few lines of code, you can start logging messages and gaining valuable insights into your application's behaviour.

Comprehensive Documentation

Dive into our comprehensive documentation to learn more about RustLogs (RLG)'s features, configuration options, and usage examples. Our docs will guide you every step of the way.

Community Support

Join our vibrant community of Rust developers and logging enthusiasts. Get help, share your experiences, and contribute to the future of RustLogs (RLG). Together, let's make logging better!

[Get Started Now](#)

divider