Welcome to the RustLogs (RLG) FAQ! Here, you'll find answers to common questions about the RustLogs library and its usage. If you don't find what you're looking for, feel free to open an issue on our GitHub repository.

RustLogs (RLG) is a Rust library offering application-level logging with intuiti ve APIs and helpful macros for ease of integration into Rust applications. It su pports multiple log levels and formats, providing a versatile logging solution. This guide covers everything from adding RustLogs to your project, creating log entries, customizing log paths, using macros for simplified logging, handling er rors, contributing to RustLogs, and finding further support.

General

What is RustLogs (RLG)?

RustLogs (RLG) is an innovative Rust library tailored for application-level logg ing, offering a straightforward, readable output format. Designed to simplify lo gging tasks in Rust applications, it brings a blend of intuitive APIs and macros to the table, facilitating the integration of logging into your codebase effortl essly. RustLogs (RLG) supports multiple log levels, structured log formats, and a wide range of output formats, making it a versatile choice for logging in Rust applications.

Why should I use RustLogs (RLG)?

RustLogs (RLG) simplifies the process of adding logging functionality to your Ru st applications. With its intuitive APIs and helpful macros, you can easily inte grate logging into your codebase and gain valuable insights into your applicatio n's behaviour. RustLogs (RLG) supports multiple log levels, structured log forma ts, and a wide range of output formats, making it a versatile choice for logging in Rust applications.

What are the key features of RustLogs (RLG)?

Supports multiple log levels:

,

,

,

,

,

,

,

,

, andStructured log formats for easy parsingProvides structured log formats that are easy to parse and filterWide compatibility with output formatsCompatible wit h various output formats, including CEF, ELF, GELF, JSON, CLF, W3C, Syslog, Apac he Access Log, Logstash, Log4j XML, NDJSON, and moreFlexible configurationOffers flexible configuration options for customizing log file path, log levels, and ou tput formatsAsynchronous logging:Supports asynchronous logging for optimal perfo rmance and responsivenessUseful macros to streamline logging tasksProvides a set of helpful macros to simplify common logging tasksdivider

Usage

How do I add RustLogs (RLG) to my Rust project?

To add RustLogs (RLG) to your Rust project, simply add the following line to you r

file:

[dependencies]

rlg = "0.0.3" How do I create a new log entry?

To create a new log entry using RustLogs (RLG), you can use the function:

```
use rlg::log::Log;
use rlg::log_format::LogFormat;
use rlg::log_level::LogLevel;
let log_entry = Log::new(
"12345",
"2023-01-01T12:00:00Z",
&LogLevel::INFO,
"MyComponent",
"This is a sample log message",
&LogFormat::JSON,
);
```

How do I log a message asynchronously?

To log a message asynchronously using RustLogs (RLG), you can use the method on a log entry:

```
tokio::runtime::Runtime::new().unwrap().block_on(async {
log_entry.log().await.unwrap();
});
```

How can I customize the log file path?

By default, RustLogs (RLG) logs to a file named "RLG.log" in the current directory. You can customize the log file path by setting the environment variable:

```
std::env::set_var("LOG_FILE_PATH", "/path/to/log/file.log");
```

divider

Macros

What macros are available in RustLogs (RLG)?

RustLogs (RLG) provides several helpful macros to simplify common logging tasks:

Creates a new log entry with specified parameters.Creates an info log with defau
lt session ID and format.Prints a log to stdout.Asynchronously logs a message to
a file.Creates a warning log.Creates an error log with default format.Sets the l
og format to CLF if not already defined.Conditionally logs a message based on th
e
feature flag.Creates a trace log.Creates a fatal log.Conditionally logs a messag
e based on a predicate.Logs a message with additional metadata. How do I use the
macro?
The
macro allows you to create a new log entry easily. Here's how you can use it:
let log = macro_log!(session_id, time, level, component, description, format); H
ow do I use the
macro?
The
macro creates an info log with default session ID and format. Here's how you can
use it:
let log = macro_info_log!(time, component, description); How do I conditionally
log a message based on a predicate?
You can use the
macro to conditionally log a message based on a predicate:
macro_log_if!(predicate, log);If the
evaluates to
, the log message will be printed using
.
divider

Error Handling

How does RustLogs (RLG) handle errors during logging operations?

Errors can occur during logging operations, such as file I/O errors or formattin g errors. The
method returns a
that indicates the outcome of the logging operation. You should handle potential errors appropriately in your code:

```
match log_entry.log().await {
Ok(_) => println!("Log entry successfully written"),
Err(err) => eprintln!("Error logging entry{}", err),
}divider
```

Contributing

How can I contribute to RustLogs (RLG)?

We welcome contributions to RustLogs (RLG)! If you'd like to contribute, please follow these steps:

Fork the RustLogs (RLG) repository on GitHub.Create a new branch for your featu re or bug fix.Make your changes and ensure that the code builds successfully.Wri te tests to cover your changes, if applicable.Run the test suite to ensure that all tests pass.Commit your changes and push them to your forked repository.Open a pull request on the main RustLogs (RLG) repository, describing your changes an d the problem they solve.We appreciate your contributions and will review your p ull request as soon as possible.

divider

Support

Where can I find more information about RustLogs (RLG)?

For more information about RustLogs (RLG), you can refer to the following resour ces:

RustLogs (RLG) Documentation RustLogs (RLG) Repository RustLogs (RLG) Crate H ow can I report a bug or request a feature?

If you encounter a bug or have a feature request for RustLogs (RLG), please open an issue on our GitHub repository. Provide as much detail as possible about the problem or feature you're requesting, and we'll do our best to address it.

How can I get help with using RustLogs (RLG)?

If you need help with using RustLogs (RLG), you can:

Check the RustLogs (RLG) Documentation for detailed usage instructions and exam ples.Search for existing issues or discussions on the RustLogs (RLG) GitHub repo sitory to see if your question has already been answered.If you can't find a so lution, feel free to open a new issue on the GitHub repository, describing your problem or question in detail. We'll do our best to assist you.We hope this FAQ has been helpful in answering your questions about RustLogs (RLG). If you have a ny further inquiries or need additional assistance, please don't hesitate to rea ch out to us. Happy logging with RustLogs (RLG)!