LizardByte

ReenigneArcher

Jun 08, 2026

ABOUT

CHAPTER

ONE

OVERVIEW

LizardByte has the full documentation hosted on Read the Docs.

This documentation is being localized on Crowdin.

Community!

Accurate translations depend on contributions from amazing community members, like you! If you would like to contribute to the localization of this documentation, please visit our Crowdin project. Thanks to all the contributors for their hard work!

1.1 About

LizardByte is developing self hosted cloud game streaming solutions. These applications are developed by volunteers in their free time. If you like our products please consider donating!

CHAPTER

TWO

SUPPORT

> **Warning**
>
> We use GitHub issues exclusively for actionable bug reports. Low-effort issues will be immediately closed. Spamming the GitHub repo or organization will result in a ban.

2.1 Getting Support

First, read the documentation! Many common questions are answered in our comprehensive docs.

Platform	Support Type	Description
Read the Docs	**Documentation**	Official documentation and guides
Discord	**Primary Support**	Real-time chat support
GitHub Discussions	**Primary Support**	Feature requests and general discourse
Reddit	**Community Support**	Community-driven support

2.2 Additional Resources

- Homepage
- Status Page
- Blog
- Roadmap

2.3 Support Guidelines

When seeking support, please:

1. **Read the documentation first** - Many common questions are already answered in docs
2. **Use the appropriate channel** - Each platform has multiple channels/categories for support, select the one that best fits your question
3. **Be courteous and respectful** - Follow community guidelines and be patient with volunteers
4. **Provide detailed information** - Include logs, error messages, and system information when relevant
5. **Use proper formatting** - Help others help you by formatting code and logs properly
6. **Do not cross post** - Please avoid posting the same question in multiple places

CHAPTER

THREE

DONATE

Want to support our work? You can donate to us via the following platforms:

If you'd like to support us without donating, please consider staring your favorite LizardByte repositories.

Thank you to all our supporters!

CHAPTER

FOUR

COMMUNITY RULES

4.1 Be Respectful

1. **Remember the human** - Be respectful to everyone and act professionally.
2. **Don't spam** - You will be kicked if you spam.
3. **Mention etiquette** - Don't mention other users unnecessarily. Especially don't mention (or DM) the developers or support staff. We read every single message.

4.2 Communication Guidelines

4. **Use appropriate channels** - Please use the correct channel for your question and stay on topic within the channel.
5. **No cross-posting** - Cross posting among multiple communication mediums or channels is time-consuming for developers and moderators. Please be respectful of our time.
6. **Be patient** - Not everyone is available all the time. Someone will respond to you when they can.

4.3 Before Asking for Help

7. **Read the documentation** - Your question may already be answered in the documentation. Also check the open issues. Documentation is specific to each project.
8. **Search first** - Search the channel before asking. Search open issues on GitHub, GitHub Discussions, and the Documentation.

4.4 Support Requests

9. **Provide your logs** - Without logs we can't help. Please do not attach screenshots of terminal output. For Discord users, please upload your complete logs using Gist (preferred method, as we cannot see pasted log files when viewing on Discord mobile).

4.5 Prohibited Content

10. **No fork discussions** - If you'd like to discuss forks of our projects, please do so wherever the fork maintainer has set up for such topics.
11. **No piracy** - Don't discuss where to get media content, ROMs, BIOS, etc. Breaking this rule will result in an instant ban!
12. **No NSFW content** - Don't post any NSFW content. Breaking this rule will result in an instant ban!

13. **No hate speech** - Don't post any hate speech. Breaking this rule will result in an instant ban!
14. **No harassment** - Don't harass other users. Breaking this rule will result in an instant ban!
15. **No doxxing** - Don't post any personal information about other users. Breaking this rule will result in an instant ban!
16. **No illegal content** - Don't post any illegal content. Breaking this rule will result in an instant ban!
17. **No phishing** - Don't post any phishing links. Breaking this rule will result in an instant ban!
18. **No scamming** - Don't post any scamming links. Breaking this rule will result in an instant ban!

CHAPTER

FIVE

CONTRIBUTOR COVENANT CODE OF CONDUCT

5.1 Our Pledge

We as members, contributors, and leaders pledge to make participation in our community a harassment-free experience for everyone, regardless of age, body size, visible or invisible disability, ethnicity, sex characteristics, gender identity and expression, level of experience, education, socio-economic status, nationality, personal appearance, race, caste, color, religion, or sexual identity and orientation.

We pledge to act and interact in ways that contribute to an open, welcoming, diverse, inclusive, and healthy community.

5.2 Our Standards

Examples of behavior that contributes to a positive environment for our community include:

- Demonstrating empathy and kindness toward other people
- Being respectful of differing opinions, viewpoints, and experiences
- Giving and gracefully accepting constructive feedback
- Accepting responsibility and apologizing to those affected by our mistakes, and learning from the experience
- Focusing on what is best not just for us as individuals, but for the overall community

Examples of unacceptable behavior include:

- The use of sexualized language or imagery, and sexual attention or advances of any kind
- Trolling, insulting or derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or email address, without their explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

5.3 Enforcement Responsibilities

Community leaders are responsible for clarifying and enforcing our standards of acceptable behavior and will take appropriate and fair corrective action in response to any behavior that they deem inappropriate, threatening, offensive, or harmful.

Community leaders have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, and will communicate reasons for moderation decisions when appropriate.

5.4 Scope

This Code of Conduct applies within all community spaces, and also applies when an individual is officially representing the community in public spaces. Examples of representing our community include using an official e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event.

5.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported to the community leaders responsible for enforcement using our support center. All complaints will be reviewed and investigated promptly and fairly.

All community leaders are obligated to respect the privacy and security of the reporter of any incident.

5.6 Enforcement Guidelines

Community leaders will follow these Community Impact Guidelines in determining the consequences for any action they deem in violation of this Code of Conduct:

5.6.1 1. Correction

Community Impact: Use of inappropriate language or other behavior deemed unprofessional or unwelcome in the community.

Consequence: A private, written warning from community leaders, providing clarity around the nature of the violation and an explanation of why the behavior was inappropriate. A public apology may be requested.

5.6.2 2. Warning

Community Impact: A violation through a single incident or series of actions.

Consequence: A warning with consequences for continued behavior. No interaction with the people involved, including unsolicited interaction with those enforcing the Code of Conduct, for a specified period of time. This includes avoiding interactions in community spaces as well as external channels like social media. Violating these terms may lead to a temporary or permanent ban.

5.6.3 3. Temporary Ban

Community Impact: A serious violation of community standards, including sustained inappropriate behavior.

Consequence: A temporary ban from any sort of interaction or public communication with the community for a specified period of time. No public or private interaction with the people involved, including unsolicited interaction with those enforcing the Code of Conduct, is allowed during this period. Violating these terms may lead to a permanent ban.

5.6.4 4. Permanent Ban

Community Impact: Demonstrating a pattern of violation of community standards, including sustained inappropriate behavior, harassment of an individual, or aggression toward or disparagement of classes of individuals.

Consequence: A permanent ban from any sort of public interaction within the community.

5.7 Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 2.1, available at https://www.contributor-covenant.org/version/2/1/code_of_conduct.html.

Community Impact Guidelines were inspired by Mozilla's code of conduct enforcement ladder.

For answers to common questions about this code of conduct, see the FAQ at https://www.contributor-covenant.org/faq. Translations are available at https://www.contributor-covenant.org/translations.

CHAPTER

SIX

CONTRIBUTING

6.1 Guidelines

Thank you for considering contributing to our open source project! We welcome all contributions, whether you're fixing a bug, improving documentation, or adding a new feature.

Before you start, please take a moment to read our code of conduct and community guidelines, which outline our expectations for respectful and inclusive behavior.

6.1.1 Getting Started

1. Fork the repository on GitHub to create a copy of the project.
2. Create a new branch for your changes. This will help if you need to create multiple pull requests, which may be out of scope from each other.
3. Make your changes, be sure to follow our code style guidelines and test your code thoroughly.
4. Document your code changes to the best of your ability. This will help other contributors understand the purpose and functionality of your changes.
5. You can also add missing documentation to any related code. Documentation is important for the maintainability of our projects.
6. When you're ready, submit a pull request using the template provided.

6.1.2 Pull Requests

Creating a Pull Request

To help us review your changes quickly and accurately, please follow these guidelines when submitting a pull request:

1. Submit your pull request against the default branch. The pull request title should follow conventional commit guidelines. If we accept the PR, it will be squashed and merged with the pull request title and number as the commit message.
2. Complete the pull request template, including any relevant details about your changes and any associated issues.
    - Be sure to use the official template. Do not use AI-generated PR summaries that completely replace our template.
    - Leave the template comments in place. They are helpful when editing the PR description.

      ```
      <!-- This is a comment -->
      ```

    - Do not delete any sections of the template, even if they do not apply to your changes.
    - Link any issues or discussions that are resolved by your changes.

```
- Closes #123
- Fixes #456
- Resolves #789
```

- Complete the checklists. It is not required to check everything, only check the items that apply to your changes.

```
- [x] This is a complete item
- [ ] This is an incomplete item
```

3. Be prepared to address any feedback or questions that may arise during the code review process.

> **Note**
>
> It is important to note that pull requests should generally address a single issue or feature to make it easier to review and test the code. In some cases, exceptions may be made. For example, if you find a typo or formatting error in a file being modified, it may be acceptable to fix it in the same pull request as a drive-by fix. However, if the pull request is already large, it may be better to create a separate pull request for the secondary fixes.

Review Process

Pull requests will be reviewed by the project maintainers to ensure code quality and consistency with project standards.

The changes requested may be minor, such as fixing a typo, adjusting formatting, or adding a comment. These may seem like small requests, but they are important to our projects as they help ensure that the code is readable, maintainable, and consistent with the rest of the project. In other cases, the requested changes may be more significant. In either case, the maintainers will provide feedback to help you improve the code.

Please keep in mind that partially complete pull requests will not be merged. Before merging, we will consider the following criteria:

- Does the code follow the style guidelines of the project?
- Does the change add value?
- Is the code well commented?
- Have documentation blocks been updated for new or modified components?
- Will the changes create issues in other scenarios?

Developers and maintainers will attempt to assist with challenging issues.

6.1.3 Code Style

Code Style Guidelines

We enforce consistent code style across our projects to improve readability and maintainability of the codebase. Here are some of the guidelines we follow:

- Each file, with few exceptions, should end with an empty line.
- In most cases, the maximum line length should not exceed 120 characters to make the code more readable.
- We use yamllint to lint our *yaml* files. You can find the configuration file here in our *.github* repository.
- We use CodeQL and SonarCloud to analyze our codebases. There will be a comment on your PR indicating if there are any issues that need to be addressed. Please address these issues if reasonable. In some cases, there may

be false positives, use your best judgment to determine if changes should be made. These tools are not perfect and not meant to override engineering judgment.

Want to address SonarCloud issues locally before committing? You can use the SonarLint plugin for your IDE.

6.1.4 AI Usage

We recognize the value of AI tools for improving code and development workflows. However, all AI-generated contributions must meet our quality standards.

Acceptable AI Usage

- Using AI to help brainstorm solutions to complex problems
- Getting assistance with syntax or language-specific implementations
- Improving documentation clarity or grammar
- Generating test cases for existing code
- Code refactoring suggestions that you carefully review and understand

Unacceptable AI Usage

- AI generation of PR description/summary, use our standard PR template
- Submitting code you don't fully understand
- Generating entire features or components without significant human oversight
- Using AI to create code without properly testing it
- Submitting content with hallucinations, errors, or inconsistencies
- Contributing code that doesn't follow our established patterns and conventions

All contributions, regardless of how they were created, must meet our quality standards. AI-generated content that contains errors, doesn't solve the problem effectively, or appears to be low-quality "slop" will be rejected immediately. You are responsible for all contributions under your name, so ensure you thoroughly review any AI-assisted work before submission.

6.1.5 Testing

Testing is a critical part of our development process, and we have automated tests and tools to ensure that our code meets the expected quality and functionality.

Code Style Tests

To ensure consistent code style, we run automated tests on pull requests. The tests that run depend on the language(s) of the repository. The following table shows the labels and the corresponding tests that will run:

Language	Checks
`c`, `c++`	clang-format, cmake-lint
`docker`	hadolint
`github-action`, `github-workflow`	actionlint
`jupyter`, `python`	flake8
`PowerShell`	PSScriptAnalyzer
`rust`	rustfmt
`shell`	shellcheck
`yaml`	yamllint

Projects may have additional checks, `eslint` for example, depending on the project's requirements.

Unit Testing

We strive to have comprehensive unit tests for our projects, but this is still a work in progress for some projects. We welcome contributions that improve test coverage and add new tests.

In general, PRs should not drastically reduce coverage percentages. If a change is big enough, tests should be implemented for it. No one knows your code as well as you do, so you are the best person to write the tests for it. We understand that not everyone may be experienced with writing tests, so please reach out if you would like some assistance.

Python Projects

All Python projects in the LizardByte organization use uv for dependency management, Python environment setup, lock files, and command execution. A Python project's `pyproject.toml` and `uv.lock` files are the source of truth for its dependencies.

Use the Python version declared by the project instead of assuming the same version across every repository. After changing Python dependencies or project metadata, update and commit the lock file with the related change.

Common uv commands:

```
uv lock --check
uv sync --frozen
uv run --locked python -m pytest
uv lock
```

Useful tips:

- Use `uv lock --check` in validation to confirm the lock file still matches `pyproject.toml`.
- Use `uv sync --frozen` in custom build or CI commands to install only from `uv.lock` without updating it.
- Use native uv integrations for hosted services when available, and add a lockfile drift check when they cannot pass `--frozen` directly.
- Use `uv run --locked <command>` when you want to run a project command without changing `uv.lock`.
- Use `uv sync --locked` locally when you want uv to fail if `uv.lock` is missing or out of date.
- Use `uv lock` only when dependency inputs changed and the lock file should be updated.
- Prefer project-specific commands from each repository's documentation when they exist.

6.1.6 Localization

LizardByte projects are used by people all over the world. When feasible, we strive to make our projects multilingual. If you are able to contribute translations, we would be grateful for your help.

CrowdIn

The translations occur on CrowdIn. Anyone is free to contribute to the localization there. The project on CrowdIn is a mono-project, meaning that all translations are done in one place. This allows for easier management of translations across projects.

Translation Basics

- The brand name *LizardByte* and project names should never be translated.
- Other brand names should never be translated. Examples include *AMD*, *Intel*, and *NVIDIA*.

CrowdIn Integration

How does it work?

When a change is made to a project's source code, CrowdIn will automatically receive the updated strings.

Once translations are updated on CrowdIn, a push gets made to the *l10n_master* branch and a PR is made against the *master* branch. Once the PR is merged, all updated translations are part of the project and will be included in the next release.

Extraction

The extraction process varies by project. Please consult the project's documentation for more information.

CrowdIn Contributors

Thank you to all the contributors who have helped with translations. Your contributions are greatly appreciated!

6.1.7 Legal

We require that all contributors sign a Contributor License Agreement (CLA) before we can merge their pull requests. If any action is required, a bot will comment on your PR with instructions. Exempt repositories are listed below.

- beautiful-jekyll-next
- build-deps
- contribkit
- pacman-repo-builder
- Sunshine
- tray
- Virtual-Gamepad-Emulation-Bus
- Virtual-Gamepad-Emulation-dotnet
- Virtual-Gamepad-Emulation-Client

We offer two types of CLAs:

- CLA for individuals
- CLA for entities

If you do not own the Copyright in the entire work of authorship submitted, you must complete the following steps:

1. Add the owner(s) as a *co-author* to a commit in the PR. See Creating a commit with multiple authors.
2. All authors must sign the CLA before it can be merged.

If you are an entity, you cannot sign the CLA through CLA-assistant. You must sign the CLA for entities. Please contact LizardByte for more information.

CHAPTER

SEVEN

REPOSITORY STANDARDS

This document outlines the standards and best practices for LizardByte repositories to ensure consistency, maintainability, and quality across all projects.

7.1 Repository Structure

7.1.1 File Organization

All repositories should follow a consistent structure:

```
repository-name/
├── .github/
│   ├── matchers/  # custom problem matchers
│   ├── workflows/
│   │   ├── _codeql.yml
│   │   └── _common-lint.yml
│   ├── dependabot.yml
│   └── semantic.yml
├── branding/  # project specific branding assets
├── docs/  # structure varies depending on the doc system used
├── src/  # main source code
├── tests/  # if there are tests
├── third-party/  # location of submodules if any
├── .gitignore
├── LICENSE
└── README.md or README.rst
```

7.1.2 Required Files

Every repository must include:

- **LICENSE**: All projects must use an appropriate license
- **README**: Primary documentation in reStructuredText (.rst) or Markdown (.md) format
- **.gitignore**: Appropriate for the project's language and framework. See GitHub's collection for starter templates.
- **.github/**: Contains GitHub-specific configurations (workflows, issue templates, etc.)

7.2 Naming Conventions

7.2.1 Repository Names

- Be descriptive and concise
- Avoid abbreviations unless widely understood

7.2.2 Branch Names

- Use descriptive names with forward slashes for organization
- Format: `type/short-description`
- Types: `feat/`, `fix/`, `docs/`, `refactor/`: see Convential Commits for a full list
- Examples: `feat/user-authentication`, `fix/memory-leak`, `docs/api-reference`

7.2.3 File and Directory Names

- Be consistent within each project
- Avoid spaces and special characters

For questions about these standards or exceptions, please open an issue in the .github repository or contact the maintainers.

CHAPTER

EIGHT

SECURITY POLICY

8.1 Supported Versions

LizardByte only supports the latest version of our software.

8.2 Reporting a Vulnerability

If you think you have found a security vulnerability in our software, please report it to us privately using the "Security" tab on the GitHub repository.

After you have submitted the vulnerability, we will review the report internally and respond to you with any questions or next steps.

If the report is accepted, we will request a CVE and create a private fork of the repository to work on the issue. Once the pull request is merged, we will publish the security advisory on the GitHub repository.

We ask that you do not discuss or disclose the vulnerability outside the advisory on GitHub until the advisory is published. This helps to protect users until a fix is available.