# Contributing to Jackett

So, you've decided you want to help make Jackett a better program for everyone. Not everyone chooses to help, so we thank you for your decision.

In order to help us make the most of your contribution please take the time to read these contributing guidelines.

These are just guidelines, not hard rules. Use your best judgment, and feel free to propose changes to this document in a pull request.

## Ways you can help

- [Getting Started](#getting-started)

- [Troubleshooting](#troubleshooting)

- [Reporting a bug](#reporting-a-bug)

- [Adding a new tracker](#adding-a-new-tracker)

- [Contributing Code](#contributing-code)

- [Setting up your environment](#setting-up-your-environment)

- [Coding style](#coding-style)

- [Getting your code accepted & pull requests](#pull-requests)

# Getting Started

Now that you've decided you want to help us make Jackett a better program the big question is: Where do you start?

Why right here of course. You can help in several ways, from finding and reporting bugs, to adding new trackers,

to fixing bugs in the program code itself. Below, we outline the steps needed to file your first bug report.

## Troubleshooting

Before you submit a bug report, it's important to make sure it's not already a known issue,

and to make sure it's a bug we can find and fix quickly.

These troubleshooting tips will help make sure your bug report is high quality and can be fixed quickly.

\*\*Update your Jackett to the latest version\*\*

Before you submit a bug-report or do any other troubleshooting, make sure your Jackett is the latest release version.

We are releasing bug fixes almost daily, so your issue may have been fixed already.

Bugs that are submitted without being on the latest version may be closed.

\*\*Error "An error occurred while sending the request: Error: TrustFailure (A call to SSPI failed, see inner exception.)"\*\*

This is often caused by missing CA certificates.

Try reimporting the certificates in this case:

- On Linux (as user root): `wget -O - https://curl.haxx.se/ca/cacert.pem | cert-sync /dev/stdin`

- On macOS: `curl -sS https://curl.haxx.se/ca/cacert.pem | cert-sync --user /dev/stdin`

\*\*Tracker isn't working\*\*

If you are experiencing an issue with a tracker, then:

- Use your browser to check you can access the site directly, and if a login is required,

check you can login and that you do not have any outstanding account issues.

- If you haven't already, try upgrading to the latest version of Jackett.

- If it is still not working for you, then a \*\*full enhanced log must be included\*\*.

\*\*Enable enhanced logging\*\*

- You can get \*enhanced\* logging with the command line switches `-t -l` or by enabling `Enhanced logging` via the web interface

(followed by clicking on the `Apply Server Settings` button).

- These enhanced logs are necessary for us to quickly track down your bug and get a fix implemented in code.

- Make sure you remove your username/password/cookies from the log files before submitting them with your issue.

- The logfiles (log.txt/updater.txt) are stored on Windows in `%ProgramData%\Jackett`, on Linux/macOS in `~/.config/Jackett/`,

and on FreeBSD in `/usr/local/jackett`.

## Reporting a Bug

Once you have your enhanced logs and you are still unable to resolve your issue yourself, now it's time to prepare to submit a bug report!

Before you submit your report, make sure you've searched open \*and\* closed bugs to see if someone's already informed us of your issue.

If your search doesn't help you fix your issue and you can't find a similar bug already listed, then you get to make a new issue.

Your issue should have the following information.

- \*\*Descriptive Title\*\* - The title of your bug should include keywords and a descriptive summary of what you're experiencing

to help others avoid duplicating your bug report

- Keywords in the title should be as follows:

- Tracker bugs should start with the tracker in brackets e.g. [\*\*AnimeBytes\*\*]

- Feature requests should start with [\*\*REQ\*\*]

- New trackers should begin with [\*\*New\*\*] and the tracker type [\*\*Public\*\*/\*\*Private\*\*/\*\*Semi-Private\*\*] e.g. \*\*[New][Public] ThePirateBay\*\*

- \*\*Environment Details\*\* - These are things like your OS version, Jackett type and version, mono/.Net-core/framework version(s).

These are asked for by the issue template when you create a new issue on GitHub.

- \*\*Steps\*\* to cause the problem, if applicable. These should be specific and repeatable.

- \*\*What happens\*\* when you take the steps and \*\*what you expected to happen\*\*

- \*\*Error messages\*\* and/or screenshots of the issue.

- The \*\*last working version\*\* if it's applicable. Tracker issues normally don't need this information.

- An attached copy of your \*\*enhanced logs\*\*. Don't forget to remove usernames/passwords/API-keys from the logs.

We'll be working on making sure these are automatically censored in the future.

- Any other \*\*relevant details\*\* you can think of. The more information we have, the quicker we can solve the problem.

## Adding a New Tracker

Jackett's framework typically allows our team and volunteering developers to implement new trackers in a couple of hours

Depending on logic complexity, there are two common ways new trackers are implemented:

1. simple [definitions](http://github.com/Jackett/Jackett/tree/master/src/Jackett.Common/Definitions) (.yml / YAML)

2. advanced (native) [indexers](http://github.com/Jackett/Jackett/tree/master/src/Jackett.Common/Indexers) (.cs / C#)

Read more about the [simple definition format](https://github.com/Jackett/Jackett/wiki/Definition-format).

# Contributing Code

While reporting the bugs is super helpful since you can't fix bugs you don't know about, they don't get fixed unless someone goes in and fixes them.

Luckily, you're a developer who wants to help us do just that. Thanks!

We really need more developers working on Jackett, no matter their skill level or walk of life.

We've developed the guide below to make sure we're all on the same page because this makes reading and fixing code much simpler, faster, and less bug-prone.

## Setting up your environment

The following guide assumes you've never worked with a Visual Studio project with GitHub before.

This will give you the minimum necessary tools to get started. There are plenty of optional tools that may help you, but we won't cover those here.

- The guide is currently only geared towards developing on Windows using Visual Studio Community 2019.

If you use something else, please add it here for others.

<details open=true> <summary> Windows </summary>

<details open=true> <summary> Visual Studio 2019 </summary>

- Install [Visual Studio Community 2019](http://visualstudio.com) for free.

- About 2GB download. 8GB installed.

- Make sure it includes the following Workloads/Components:

- .Net Desktop Development

- .Net Core Cross-Platform Development

- GitHub extension for Visual Studio

- [Connect and synchronize your forked repository to Visual Studio](https://doc.fireflymigration.com/working-with-github-fork-in-visual-studio.html)

- Open `Tools -> NuGet Package Manager -> Package Manager Console`

- From the PMC, run `dotnet tool install -g dotnet-format` and `dotnet restore`

- Run `Build -> Rebuild Solution` to restore NuGet packages

- Ensure `Jackett.Server` is the Startup Project (instead of `Jackett.Service`), and the Run Target (instead of `IIS Express`)

</details>

</details>

## Coding Style

Now that you're ready to code, it's time to teach you our style guidelines. This style guide helps our code stay readable and bug-free.

You can see the full details in the [Editor Config](.editorconfig) file.

Running `dotnet format` from the Package Manager Console will apply the style guide to the solution and is required before any pull request will be accepted.

- Whitespace

- Indenting is done with 4 spaces

- No whitespace at the end of lines

- All files have a final newline

- Unix style new lines for committed code

- Spaces around all non-unary operators

- Braces

- Opening brace on its own line

- Single line statements do not use braces

- If any part of an `if ... else if ... else` block needs braces, all blocks will use braces

- Naming

- `interface` names begin with I and are `PascalCase`

- `private` variables begin with \_ and are `camelCase`

- `private static` variables begin with s\_ and are `camelCase`

- local variables are `camelCase`

- `async` function names end with Async

- all others are `PascalCase`

- Others

- Prefer `var` for declarations

- Prefer modern language enhancements (C#7, C#8 features)

- switch expressions

- range operator

- using statements

- `default` over `default(T)`

- Prefer conditional access `?.` and null coalescing `??` over null checks

- Prefer pattern matching

- Prefer expression bodies

- Avoid `this` qualifier

- `using` statements go outside namespace declaration and are sorted:

- `using System`

- `using System.\*` alphabetically

- all others alphabetically

- Prefer explicit variable modifiers: `private`, `public`, `protected`

- Prefer `readonly` and `const` variables when appropriate

## Pull Requests

At this point, you've found the bug, fixed it, tested that the bug is gone, and you haven't broken anything else in the process.

Now it's time to share your code with everyone else so we can all enjoy a better version of the program.

Here's what you need to do to give your pull request the best chance at a timely review and maximize that it will be accepted.

- Make sure your code follows GitHub and Jackett's standards and practices.

- Your changes should be made in a new branch based on `master` not directly on your `master` branch

- Your commit messages should start with a capital letter, be in the singular imperative voice, and do not end with punctuation marks, e.g.:

- Fix login handling for xxx tracker

- Add feature yyy

- Remove dead tracker fff

- Run `dotnet format` from the Package Manager Console (found in `Tools -> NuGet Package Manager` or `View -> Other Windows`)

- If your branch falls out of sync and has merge conflicts with the Jackett official `master`

[rebase](https://mohitgoyal.co/2018/04/18/working-with-git-and-visual-studio-use-git-rebase-inside-visual-studio/) your fix before submission.

- If you deleted, moved, or renamed any files/folders, be sure to add the old file/folder path to the appropriate array in `Jacket.Updater/Program.cs`

- If you added or renamed a tracker, update the README to include the new name

- [Squash your local commits](https://github.com/spottedmahn/my-blog/issues/26)

- Push your commit branch to your fork on GitHub.

- Create your Pull Request

- You can do this from the GitHub website or from the GitHub window in Visual Studio.

- Give your Pull Request a descriptive title

- Include keywords like `[New Tracker]` or `[Feature]` at the beginning of the title

- Include any open tickets this Pull Request should fix in the description. \*\*Do not\*\* put ticket numbers in the title.

We will be by when we can to review your Pull Request.