# Instructions for Logging Issues

## 1. Search for Duplicates

[Search the existing issues](https://github.com/lukeautry/tsoa/search?type=Issues) before logging a new one.

Some search tips:

- \_Don't\_ restrict your search to only open issues. An issue with a title similar to yours may have been closed as a duplicate of one with a less-findable title.

- Check for synonyms. For example, if your bug involves an interface, it likely also occurs with type aliases or classes.

- Search for the title of the issue you're about to log. This sounds obvious but 80% of the time this is sufficient to find a duplicate when one exists.

- Read more than the first page of results. Many bugs here use the same words so relevancy sorting is not particularly strong.

- If you have a crash, search for the first few topmost function names shown in the call stack.

## 2. Did you find a bug?

Please follow the issue template and include as much information as you can. This increases the speed of a fix being deployed because we need all of the information we can to be able to reproduce the bug first.

## 3. Do you have a suggestion?

We also accept suggestions in the issue tracker.

- Be sure to [search](https://github.com/lukeautry/tsoa/search?type=Issues) first for a duplicate.

- Please ensure that your suggestion adheres to the [goals](https://github.com/lukeautry/tsoa#goal) and [philosophy](https://github.com/lukeautry/tsoa#philosophy) of the project

# Instructions for Contributing Code

## General

If you have a bugfix or new feature that you would like to contribute to tsoa, please find or open an issue about it first. Talk about what you would like to do. It may be that somebody is already working on it, or that there are particular issues that you should know about before implementing the change.

We enjoy working with contributors to get their code accepted. There are many approaches to fixing a problem and it is important to find the best approach before writing too much code.

In situations where the code is very hard to extend, we may want to refactor modules or redesign certain aspects of the project. The new design should be proposed and agreed upon in the attached issue. While we strongly value enhancements to code quality, significant refactoring should usually be reserved for situations where the non-refactored version of code would be too difficult to work with.

## Contributing bug fixes

Your pull request should include a link to the bug that you are fixing. If you've submitted a PR for a bug, please post a comment in the bug to avoid duplication of effort.

## Contributing features

Features (things that add new or improved functionality to tsoa) may be accepted, but will need to \*\*first\*\* be approved by having the (["help wanted" label](https://github.com/lukeautry/tsoa/issues?q=is%3Aopen+is%3Aissue+label%3A%22help+wanted%22) by a tsoa project maintainer) in the suggestion issue.

Design changes will not be accepted at this time. If you have a design change proposal, please log a suggestion issue.

## Information on the types of tests in tsoa

### unit tests

You will find tests in `tests\unit` that test specific functions or classes within tsoa. If possible, it's ideal to add test coverage to these files since you can provide more detailed assertions about specific scenarios. These also run much more quickly. However, in some cases the only way to test tsoa (like in the case of reading a controller file) is run an integrative test. Read below:

### integrative tests

These tests read controller files (just like tsoa does for real tsoa users) (via the `tests\prepare.ts` file) and then execute API tests against the generated routes. This allows tsoa contributors to validate that all of the code is connected well and is performing the desired function. These tests take longer to run but achieve higher levels of quality assurance. A good mix of integrative tests and unit tests will surely result in a faster review process for any contribution.

## Housekeeping

Your pull request should:

- Include a description of what your change intends to do

- Be a child commit of a reasonably recent commit in the \*\*master\*\* branch

- Requests need not be a single commit, but should be a linear sequence of commits (i.e. no merge commits in your PR)

- It is desirable, but not necessary, for the tests to pass at each commit

- Have clear commit messages

- e.g. "Minor refactor in goToTypeDefinition", "Fix iterated type in for-await-of", "Add test for preserveWatchOutput on command line"

- Include adequate tests

- At least one test should fail in the absence of your non-test code changes. If your PR does not match this criteria, please specify why

- Tests should include reasonable permutations of the target fix/change

- Include baseline changes with your change

- All changed code must have some degree of code coverage (either integrative or unit level)

- To avoid line ending issues, set `autocrlf = input` and `whitespace = cr-at-eol` in your git configuration