# Contributing Code

Before submitting any contributions, it's worth your while to read the [Productivity Power Tools Roadmap](https://github.com/Microsoft/VS-PPT/wiki/Roadmap) to understand our direction. Our bar for accepting contributions is high at this point, since we're focused for now primarily on quality and infrastructure improvements, along with bug reports and suggestions.

The Productivity Power Tools project follows the [.NET Framework developer guidelines](https://github.com/dotnet/corefx/wiki). The team enforces this by regularly running the [.NET code formatter tool](https://github.com/dotnet/codeformatter) on the code base. Contributors should ensure they follow these guidelines when making submissions.

Here are the guidelines on pull requests:

\* Contributions to the master branch beyond the level of a bug fix are reserved for Microsoft PPT members at this stage.

\* Only contributions against the master branch will be accepted. Authors submitting pull requests that target experimental feature branches or release branches will likely be asked target their pull request at the master branch.

\* Pull requests that do not merge easily with the tip of the master branch will be declined. The author will be asked to merge with tip and update the pull request.

\* Submissions must meet functional and performance expectations, including scenarios for which the team doesn’t yet have open source tests. This means you may be asked to fix and resubmit your pull request against a new open test case if it fails one of these tests.

\* Submissions must follow the [.NET Foundation Coding Guidelines](https://github.com/dotnet/corefx/wiki)

\* Contributors must sign the [Microsoft Contribution License Agreement](https://cla.microsoft.com/)

When you are ready to proceed with making a change, get set up to [build the code](https://github.com/Microsoft/VS-PPT/wiki/Building,-Testing-and-Debugging-the-Sources) and familiarize yourself with our workflow and our coding conventions. These two blogs posts on contributing code to open source projects are good too: [Open Source Contribution Etiquette by Miguel de Icaza](http://tirania.org/blog/archive/2010/Dec-31.html) and [Don’t “Push” Your Pull Requests by Ilya Grigorik](https://www.igvita.com/2011/12/19/dont-push-your-pull-requests/).

If you haven't already done so, you'll need to sign the [Microsoft Contributor License Agreement (CLA)](https://cla.microsoft.com) before submitting your pull request. To complete the CLA, submit a request via the form and electronically sign the CLA when you receive the email containing the link to the document. You need to complete the CLA only once to cover all Microsoft GitHub projects.

# Developer Workflow

1. Work item is assigned to a developer during the triage process

2. Both members of the Visual Studio PPT team and external contributors are expected to do their work in a local fork and submit code for consideration via a pull request.

3. When the pull request process deems the change ready it will be merged directly into the tree.

# Creating New Issues

Please follow these guidelines when creating new issues in the issue tracker:

\* Use a descriptive title that identifies the issue to be addressed or the requested feature.

\* Do not set any bug fields other than Impact.

\* Specify a detailed description of the issue or requested feature.

\* For bug reports, please also:

- Describe the expected behavior and the actual behavior. If it is not self-evident such as in the case of a crash, provide an explanation for why the expected behavior is expected.

- Provide example code that reproduces the issue.

- Specify any relevant exception messages and stack traces.

- Subscribe to notifications for the created issue in case there are any follow up questions.

# Coding Conventions

\* Use the coding style outlined in the [.NET Foundation Coding Guidelines](https://github.com/dotnet/corefx/wiki)

\* Use plain code to validate parameters at public boundaries. Do not use Contracts or magic helpers.

```

if (argument == null)

{

throw new ArgumentNullException(nameof(argument));

}

```

\* Use ```Debug.Assert()``` for checks not needed in retail builds. Always include a “message” string in your assert to identify failure conditions. Add assertions to document assumptions on non-local program state or parameter values.

# Code Formatter

The PPT team regularly runs the [.NET code formatter tool](https://github.com/dotnet/codeformatter) to ensure the code base maintains a consistent style over time.