Contributors

============

Submitting an Issue? See the [Submitting Issues](https://github.com/chocolatey/chocolateygui#submitting-issues) section in the [README](https://github.com/chocolatey/chocolateygui/blob/master/README.md#submitting-issues).

The process for contributions is roughly as follows:

## Prerequisites

\* Ensure you have signed the Contributor License Agreement (CLA) - without this we are not able to take contributions that are not trivial.

\* [Sign the Contributor License Agreement.](https://www.clahub.com/agreements/chocolatey/ChocolateyGUI)

\* You must do this for each Chocolatey project that requires it.

\* If you are curious why we would require a CLA, we agree with Julien Ponge - take a look at his [post](https://julien.ponge.org/blog/in-defense-of-contributor-license-agreements/).

## Contributing Process

### Get Buyoff Or Find Open Community Issues/Features

\* Through GitHub, or through the [mailing list](https://groups.google.com/forum/#!forum/chocolateygui) (preferred), you talk about a feature you would like to see (or a bug), and why it should be in Chocolatey.

\* If approved through the mailing list, ensure the accompanying GitHub issue is created with information and a link back to the mailing list discussion.

\* Once you get a nod from one of the [Chocolatey Team](https://github.com/chocolatey?tab=members), you can start on the feature.

\* Alternatively, if a feature is on the issues list with the [community tag](https://github.com/chocolatey/chocolateygui/labels/Community), it is open for a patch. You should comment that you are signing up for it on the issue so someone else doesn't also sign up for the work.

### Set Up Your Environment

\* You create, or update, a fork of chocolatey/chocolateygui under your GitHub account.

\* From there you create a branch named specific to the feature.

\* In the branch you do work specific to the feature.

\* Please also observe the following:

\* No reformatting

\* No changing files that are not specific to the feature

\* More covered below in the \*\*Prepare commits\*\* section.

\* Test your changes and please help us out by updating and implementing some automated tests. It is recommended that all contributors spend some time looking over the tests in the source code. You can't go wrong emulating one of the existing tests and then changing it specific to the behavior you are testing.

\* Please do not update your branch from the master unless we ask you to. See the responding to feedback section below.

### Prepare Commits

This section serves to help you understand what makes a good commit.

A commit should observe the following:

\* A commit is a small logical unit that represents a change.

\* Should include new or changed tests relevant to the changes you are making.

\* No unnecessary whitespace. Check for whitespace with `git diff --check` and `git diff --cached --check` before commit.

\* You can stage parts of a file for commit.

A commit message should observe the following:

\* The first line of the commit message should be a short description around 50 characters in length and be prefixed with the GitHub issue it refers to with parentheses surrounding that. If the GitHub issue is #25, you should have `(GH-25)` prefixed to the message.

\* If the commit is about documentation, the message should be prefixed with `(doc)`.

\* If it is a trivial commit or one of formatting/spaces fixes, it should be prefixed with `(maint)`.

\* After the subject, skip one line and fill out a body if the subject line is not informative enough.

\* The body:

\* Should indent at `72` characters.

\* Explains more fully the reason(s) for the change and contrasts with previous behavior.

\* Uses present tense. "Fix" versus "Fixed".

A good example of a commit message is as follows:

```

(GH-7) Installation Adds All Required Folders

Previously the installation script worked for the older version of

Chocolatey. It does not work similarly for the newer version of choco

due to location changes for the newer folders. Update the install

script to ensure all folder paths exist.

Without this change the install script will not fully install the new

choco client properly.

```

### Submit Pull Request (PR)

Prerequisites:

\* You are making commits in a feature branch.

\* All specs should be passing.

Submitting PR:

\* Once you feel it is ready, submit the pull request to the `chocolatey/chocolateygui` repository against the ````develop```` branch ([more information on this can be found here](https://help.github.com/articles/creating-a-pull-request)).

\* In the pull request, outline what you did and point to specific conversations (as in URLs) and issues that you are are resolving. This is a tremendous help for us in evaluation and acceptance.

\* Once the pull request is in, please do not delete the branch or close the pull request (unless something is wrong with it).

\* One of the Chocolatey Team members, or one of the committers, will evaluate it within a reasonable time period (which is to say usually within 2-4 weeks). Some things get evaluated faster or fast tracked. We are human and we have active lives outside of open source so don't fret if you haven't seen any activity on your pull request within a month or two. We don't have a Service Level Agreement (SLA) for pull requests. Just know that we will evaluate your pull request.

### Respond to Feedback on Pull Request

We may have feedback for you to fix or change some things. We generally like to see that pushed against the same topic branch (it will automatically update the Pull Request). You can also fix/squash/rebase commits and push the same topic branch with `--force` (it's generally acceptable to do this on topic branches not in the main repository, it is generally unacceptable and should be avoided at all costs against the main repository).

If we have comments or questions when we do evaluate it and receive no response, it will probably lessen the chance of getting accepted. Eventually this means it will be closed if it is not accepted. Please know this doesn't mean we don't value your contribution, just that things go stale. If in the future you want to pick it back up, feel free to address our concerns/questions/feedback and reopen the issue/open a new PR (referencing old one).

Sometimes we may need you to rebase your commit against the latest code before we can review it further. If this happens, you can do the following:

\* `git fetch upstream` (upstream would be the mainstream repo or `chocolatey/chocolateygui` in this case)

\* `git checkout develop`

\* `git rebase upstream/develop`

\* `git checkout your-branch`

\* `git rebase develop`

\* Fix any merge conflicts

\* `git push origin your-branch` (origin would be your GitHub repo or `your-github-username/chocolateygui` in this case). You may need to `git push origin your-branch --force` to get the commits pushed. This is generally acceptable with topic branches not in the mainstream repository.

The only reasons a pull request should be closed and resubmitted are as follows:

\* When the pull request is targeting the wrong branch (this doesn't happen as often).

\* When there are updates made to the original by someone other than the original contributor. Then the old branch is closed with a note on the newer branch this supersedes #github\_number.

## Other General Information

If you reformat code or hit core functionality without an approval from a person on the Chocolatey Team, it's likely that no matter how awesome it looks afterwards, it will probably not get accepted. Reformatting code makes it harder for us to evaluate exactly what was changed.

If you do these things, it will be make evaluation and acceptance easy. Now if you stray outside of the guidelines we have above, it doesn't mean we are going to ignore your pull request. It will just make things harder for us. Harder for us roughly translates to a longer SLA for your pull request.