* Contributions
  + [Report Bugs](#user-content-fix-bugs)
  + Fix Bugs
  + Issue reporting and resolution process
  + Implement Features
  + Improve Documentation
* Pull Request Guidelines
* TCIV3 Git Branches
* Documentation
* How to sync your fork
* How to rebase PR

Contributions

Contributions are welcome and are greatly appreciated! Every little bit helps, and credit will always be given.

This document aims to explain the subject of contributions if you have not contributed to any Open-Source project, but it will also help people who have contributed to other projects learn about the rules of that community.

Fix Bugs

Look through the GitHub issues for bugs. Anything is open to whoever wants to implement it. Bug report [template](https://github.com/mickconley/TCIV3-Implementation/blob/Report-Bugs/TCI-Reported-Bugs-v1.xlsx).

Issue reporting and resolution process

An unusual element of the TCI project is that you can open a pull request (PR) to fix an issue or make an enhancement, without needing to open an issue first. This is intended to make it as easy as possible to contribute to the project.

The TCI project uses a set of labels for tracking and triaging issues, as well as a set of priorities and milestones to track how and when the enhancements and bug fixes make it into an TCIV3 release. This is documented as part of the Issue reporting and resolution process,

Implement Features

Any feature request shall be put in the bug reporting [template](https://github.com/mickconley/TCIV3-Implementation/blob/Report-Bugs/TCI-Reported-Bugs-v1.xlsx) for now. It will be reviewed by the TCI community at large for any volunteers to implement it.

Improve Documentation

TCI docs could always use better documentation, whether as part of the official TCI docs, in docstrings, docs/\*.rst or even on the web as blog posts or articles.

[Pull Request Guidelines](https://github.com/Wikia/guidelines/blob/master/Process/PullRequests.md)

Before you submit a pull request (PR) from your forked repo, check that it meets these guidelines:

* Include tests, either as unit tests, system test or both, to your pull request.
* Rebase your fork and resolve all conflicts.
* When merging PRs, Committer will use **Squash and Merge** which means then your PR will be merged as one commit, regardless of the number of commits in your PR. During the review cycle, you can keep a commit history for easier review, but if you need to, you can also squash all commits to reduce the maintenance burden during rebase.
* Add an [.. License](http://www.apache.org/legal/src-headers.html) header to all new files.
* If your pull request adds functionality, make sure to update the docs as part of the same PR. Doc string is often sufficient. Make sure to follow the Sphinx compatible standards.
* Run tests locally before opening PR.

.

TCIV3 Git Branches

All new development in TCIV3 happens in the***ActiveCode*** branch. All PRs should target that branch.

How to sync your fork

When you have your fork, you should periodically synchronize the main of your fork with the .. TCIV3 ActiveCode. In order to do that you can git pull --rebase to your local git repository from ..your brance remote and push the ActiveCode (often with --force to your fork). There is also an easy way to sync your fork in GitHub's web UI with the Fetch upstream feature. This will force-push the ActiveCode branch from ../TCIV3 to the main branch in your fork.

How to rebase PR

A lot of people are unfamiliar with the rebase workflow in Git, but we think it is an excellent workflow, providing a better alternative to the merge workflow. We've therefore written a short guide for those who would like to learn it.

As opposed to the merge workflow, the rebase workflow allows us to clearly separate your changes from the changes of others. It puts the responsibility of rebasing on the author of the change. It also produces a "single-line" series of commits on the main branch. This makes it easier to understand what was going on and to find reasons for problems (it is especially useful for "bisecting" when looking for a commit that introduced some bugs).

First, we suggest you read about the rebase workflow here: [Merging vs. rebasing](https://www.atlassian.com/git/tutorials/merging-vs-rebasing). This is an excellent article that describes all the ins/outs of the rebase workflow. I recommend keeping it for future reference.

The goal of rebasing your PR on top of ../ActiveCode is to "transplant" your change on top of the latest changes that are merged by others. It also allows you to fix all the conflicts that arise as a result of other people changing the same files as you and merging the changes to ../ActiveCode.