\_When in doubt, make an issue. If any of these instructions are unclear, make an issue to discuss your issue or suggestion.\_

### Issue Guidelines

Please follow these simple requirements before posting a bug report:

- TShock version number

- Any stack traces that may have happened when the issue occurred

- How to reproduce the issue

- Screenshots of the issue (if applicable)

### TShock Additions

If something is better suited to be a plugin for TShock, rather than a TShock core feature, it should not be added! Project scope is at times questionable, though, so create an issue on Github for discussion first. If an issue is completely outside of the scope of TShock, it will be made clear in that issue what it is.

\_If you are confused, make a suggestion. We will determine scope and relevance for you.\_

\_If a person makes a suggestion in Discord, capture the suggestion as a Github issue. If a suggestion crops up on the forums, make a Github issue to capture it. If you want, direct the user to make a suggestion on Github, but set an alarm/timer/reminder so that if they don't know how to use Github or they don't have an account, an issue is still made and discussed. Make it clear that the issue is a surrogate issue for a suggestion from Discord/the forums too.\_

### Pull Request Dev Guidelines

These guidelines are for all contributors.

\* Create an issue first to suggest an improvement or feature addition to TShock.

\* Active developers will then give a go/no go for implementation. This is scope related: if an issue is within the scope of TShock, it will be tagged 'Contribution Wanted.'

\* After 'Contribution Wanted' has been added, an issue should be considered workable in a pull request fashion.

\* If you, as a developer, want to claim an issue for a PR, as soon as possible start work and note that in both the original issue and the new PR. The 'Contribution Wanted' tag will remain but the active PR will become the center for discussion for your implementation.

\* If a TShock core developer takes an issue, they'll be assigned to the issue. If your issue was taken by a TShock developer and you were actively developing it in a PR, you should \_make it clear as soon as possible that a process error has been made\_ so that the your development resources and our development resources aren't wasted.

\* Please send a pull request with at least a sentence description and something meaningful as the title, not just the issue number you're fixing.

\_The tag indicates an issue should be implemented. If an issue has a developer assigned, it indicates that they're working on it. When in doubt, ask where an issue is before starting work (so you don't waste time)!\_

Even if you have write access to the repository, follow [Github flow](https://guides.github.com/introduction/flow/) when sending commits. Don't send commits directly to either ```master``` or ```general-devel``` unless those commits modify either the deploy scripts or non-code components. If it compiles, follow Github Flow.

Required:

- Push code to the general-devel branch. Do not push it anywhere else.

- Use tabs, not spaces.

- Use UpperCamelCase for public function names.

- Prior to developing, make sure your clone is up to date with general-devel. This means that we don't get merge commits in your pull request.

- Document all code with public access at minimum. Use the ```<param name="Name">Description</param>``` style -- do not repeat the param type or name in the xml comment field.

- Document functions that can throw exceptions with ```<exception cref="NameOfException"/>```.

- Use explicit type declaration (```int, long, float```) when the type is not easily inferred (such as return types).

- Use implicit type declaration (```var```) when the type is easily inferred (such as private lists, temporary values, etc).

- Use Microsoft code conventions on variable naming (including ```\_name``` for private members).

- When using static methods on primitives, use the CLR type. E.g. ```String.Format``` instead of ```string.Format```.

- Always use properties, not public fields.

- Document deprecations and fail compilation if they're included with ```[Obsolete("Use blah instead of blahx...", true)]```.

- Update the `CHANGELOG.md` file.

### Dev Team Guidelines

These guidelines are to be followed by all developers with commit level access to this repository:

- Prior to posting any version on the website, you must tick the version in AssemblyInfo.cs. This is the versioning formula:

- Major.Minor.Revision

- Do not release any development builds on the forums without consulting another developer first.

- This is not a professional software product. Your results may vary with code quality, buginess, etc. Do not complain about something -- just fix it and move on.

- \_\_Do not force push the repo\_\_, or you will be removed.

- \_\_Do not revert commits\_\_, unless you have sign-off from one other developer (the two-man rule), or you will be removed.

- Deprecation of a function guarantees that it will exist in the given release, but may be removed, at any time following, from subsequent releases without further warning. Warning should be given in the release thread prior to it going live.

- Updates should be discussed, via pull request of a version tick, prior to release. Only after consensus from active contributing community members has been given can a release happen.

- Breaking API changes (excluding removal of already deprecated and warned code) should be forewarned with a one week notice on the forums, which may be given at any time prior to release (as soon as a pull request for a version push has been made, an update can be warned).

- Push less compiler warnings than you started with.

- Push more documentation than you started with.

#### Pull Request Acceptance Guidelines

- Don't ruin someone's first time sending a pull request. Be civil and welcoming.

- Don't accept untested pull requests from the outside world. Bamboo and Travis will at least make sure that something compiles, but actual code and execution tests are required.

- Pull request acceptance from internal contributors (anyone with write access) requires only one other approval to merge.

- Pull request acceptance from external contributors (anyone without write access) requires the [two-man rule](https://en.wikipedia.org/wiki/Two-man\_rule) to be followed. If another man/woman/child in the two-man rule cannot be found within seven days, then this requirement is exempted.