

# Release Process

## Pre-Release

Update go.mod for submodules to depend on the new release which will happen in the next step.

1. Run the pre-release script. It creates a branch `pre_release_<new tag>` that will contain all release changes.

```
./pre_release.sh -t <new tag>
```

2. Verify the changes.

```
git diff main
```

This should have changed the version for all modules to be `<new tag>`.

3. Update the [Changelog](#).

- Make sure all relevant changes for this release are included and are in language that non-contributors to the project can understand. To verify this, you can look directly at the commits since the `<last tag>`.

```
git --no-pager log --pretty=oneline "<last tag>..HEAD"
```

- Move all the `Unreleased` changes into a new section following the title scheme ( `[<new tag>] - <date of release>` ).
  - Update all the appropriate links at the bottom.
4. Push the changes to upstream and create a Pull Request on GitHub. Be sure to include the curated changes from the [Changelog](#) in the description.

## Tag

Once the Pull Request with all the version changes has been approved and merged it is time to tag the merged commit.

**IMPORTANT:** It is critical you use the same tag that you used in the Pre-Release step! Failure to do so will leave things in a broken state.

**IMPORTANT:** [There is currently no way to remove an incorrectly tagged version of a Go module](#). It is critical you make sure the version you push upstream is correct. [Failure to do so will lead to minor emergencies and tough to work around](#).

1. Run the tag.sh script using the `<commit-hash>` of the commit on the main branch for the merged Pull Request.

```
./tag.sh <new tag> <commit-hash>
```

2. Push tags to the upstream remote (not your fork: `github.com/open-telemetry/opentelemetry-go.git` ). Make sure you push all sub-modules as well.

```
git push upstream <new tag>
git push upstream <submodules-path/new tag>
...
```

## Release

Finally create a Release for the new `<new tag>` on GitHub. The release body should include all the release notes from the Changelog for this release. Additionally, the `tag.sh` script generates commit logs since last release which can be used to supplement the release notes.

## Verify Examples

After releasing verify that examples build outside of the repository.

```
./verify_examples.sh
```

The script copies examples into a different directory removes any `replace` declarations in `go.mod` and builds them. This ensures they build with the published release, not the local copy.

## Contrib Repository

Once verified be sure to [make a release for the contrib repository](#) that uses this release.