# Guidelines For Contributing

## Before submitting an issue

- If you're not using the latest master to generate API clients or server stubs, please give it another try by pulling the latest master as the issue may have already been addressed. Ref: [Getting Started](https://github.com/openapitools/openapi-generator#getting-started)

- Search the [open issue](https://github.com/openapitools/openapi-generator/issues) and [closed issue](https://github.com/openapitools/openapi-generator/issues?q=is%3Aissue+is%3Aclosed) to ensure no one else has reported something similar before.

- File an [issue ticket](https://github.com/openapitools/openapi-generator/issues/new) by providing all the required information. Failure to provide enough detail may result in slow response from the community.

- Test with the latest master by building the JAR locally to see if the issue has already been addressed.

- You can also make a suggestion or ask a question by opening an "issue".

## Before submitting a PR

- Search the [open issue](https://github.com/openapitools/openapi-generator/issues) to ensure no one else has reported something similar and no one is actively working on similar proposed change.

- If no one has suggested something similar, open an ["issue"](https://github.com/openapitools/openapi-generator/issues) with your suggestion to gather feedback from the community.

- If you're adding a new option to a generator, please consider using the `-t` option with customized templates instead or start a discussion first by opening an issue as we want to avoid adding too many options to the generator.

- It's recommended to \*\*create a new git branch\*\* for the change so that the merge commit message looks nicer in the commit history.

## How to contribute

### git

If you're new to git, you may find the following FAQs useful:

https://github.com/openapitools/openapi-generator/wiki/FAQ#git

### Branches

Please file the pull request against the correct branch, e.g. `master` for non-breaking changes. See the [Git Branches](https://github.com/OpenAPITools/openapi-generator/wiki/Git-Branches) page for more information.

### Code generators

All the code generators can be found in [modules/openapi-generator/src/main/java/org/openapitools/codegen/languages](https://github.com/openapitools/openapi-generator/tree/master/modules/openapi-generator/src/main/java/org/openapitools/codegen/languages)

If you want to add a new generator, follow the [new-generator](https://openapi-generator.tech/docs/new-generator) guide.

### Templates

All the templates ([mustache](https://mustache.github.io/)) can be found in [modules/openapi-generator/src/main/resources](https://github.com/openapitools/openapi-generator/tree/master/modules/openapi-generator/src/main/resources).

For a list of variables available in the template, please refer to this [page](https://github.com/openapitools/openapi-generator/wiki/Mustache-Template-Variables)

### Style guide

Code change should conform to the programming style guide of the respective languages:

- Ada: https://en.wikibooks.org/wiki/Ada\_Style\_Guide/Source\_Code\_Presentation

- Android: https://source.android.com/source/code-style.html

- Bash: https://github.com/bahamas10/bash-style-guide

- C#: https://msdn.microsoft.com/en-us/library/vstudio/ff926074.aspx

- C++: https://google.github.io/styleguide/cppguide.html

- C++ (Tizen): https://wiki.tizen.org/Native\_Platform\_Coding\_Idiom\_and\_Style\_Guide#C.2B.2B\_Coding\_Style

- Clojure: https://github.com/bbatsov/clojure-style-guide

- Dart: https://www.dartlang.org/guides/language/effective-dart/style

- Elixir: https://github.com/christopheradams/elixir\_style\_guide

- Eiffel: https://www.eiffel.org/doc/eiffel/Coding%20Standards

- Erlang: https://github.com/inaka/erlang\_guidelines

- Haskell: https://github.com/tibbe/haskell-style-guide/blob/master/haskell-style.md

- Java: https://google.github.io/styleguide/javaguide.html

- JavaScript: https://github.com/airbnb/javascript/

- Kotlin: https://kotlinlang.org/docs/reference/coding-conventions.html

- Groovy: http://groovy-lang.org/style-guide.html

- Go: https://github.com/golang/go/wiki/CodeReviewComments

- ObjC: https://github.com/NYTimes/objective-c-style-guide

- Perl: http://perldoc.perl.org/perlstyle.html

- PHP: https://github.com/php-fig/fig-standards/blob/master/accepted/PSR-12-extended-coding-style-guide.md

- PowerShell: https://msdn.microsoft.com/en-us/library/dd878270(v=vs.85).aspx

- Python: https://www.python.org/dev/peps/pep-0008/

- R: https://google.github.io/styleguide/Rguide.xml

- Ruby: https://github.com/bbatsov/ruby-style-guide

- Rust: https://github.com/rust-lang-nursery/fmt-rfcs/blob/master/guide/guide.md (the default [rustfmt](https://github.com/rust-lang-nursery/rustfmt) configuration)

- Scala: http://docs.scala-lang.org/style/

- Swift: [Apple Developer](https://developer.apple.com/library/prerelease/ios/documentation/Swift/Conceptual/Swift\_Programming\_Language/TheBasics.html)

- TypeScript: https://github.com/Microsoft/TypeScript/wiki/Coding-guidelines

For other languages, feel free to suggest.

You may find the current code base not 100% conform to the coding style and we welcome contributions to fix those.

For [Vendor Extensions](https://github.com/OAI/OpenAPI-Specification/blob/master/versions/2.0.md#vendorExtensions), please follow the naming convention below:

- For general vendor extension, use lower case and hyphen. e.g. `x-is-unique`, `x-content-type`

- For language-specified vendor extension, put it in the form of `x-{lang}-{extension-name}`. e.g. `x-objc-operation-id`, `x-java-feign-retry-limit`

- For a list of existing vendor extensions in use, please refer to https://github.com/openapitools/openapi-generator/wiki/Vendor-Extensions. If you've added new vendor extensions as part of your PR, please update the wiki page.

### Testing

To add test cases (optional) covering the change in the code generator, please refer to [modules/openapi-generator/src/test/java/org/openapitools/codegen](https://github.com/openapitools/openapi-generator/tree/master/modules/openapi-generator/src/test/java/org/openapitools/codegen)

To test the templates, please perform the following:

- Update the Petstore sample by running the shell scripts under the `bin` folder. For example, run `./bin/generate-samples.sh .

/bin/configs/python\*` to update the Python-related samples under [`samples`](https://github.com/openapitools/openapi-generator/tree/master/samples). For Windows, please install [GIT bash](https://gitforwindows.org/). (If you find that there are new files g

enerated or unexpected changes as a result of the update, that's not unusual as the test cases are added to the OpenAPI spec fro

m time to time. If you've questions or concerns, please open a ticket to start a discussion)

- During development it can be helpful to quickly regenerate the samples without recompiling all of openapi-generator, e.g. when you have only updated the mustache templates. This can be done by passing the `-t` parameter: `-t modules/openapi-generator/src/main/resources/python`.

- Run the tests in the sample folder using maven `mvn integration-test -f /path/to/pom.xml`, e.g. `mvn integration-test -f samples/client/petstore/python/pom.xml`. (some languages may not contain unit testing for Petstore and we're looking for contribution from the community to implement those tests)

- Finally, git commit the updated samples files: `git commit -a`

(`git add -A` if added files with new test cases)

- For new test cases, please add to the [Fake Petstore spec](https://github.com/OpenAPITools/openapi-generator/blob/master/modules/openapi-generator/src/test/resources/3\_0/petstore-with-fake-endpoints-models-for-testing.yaml)

To start the CI tests, you can:

- Run `mvn verify -Psamples`, assuming you have all the required tools installed to run tests for different languages.

- Leverage http://travis-ci.org to run the CI tests by adding your own openapi-generator repository.

- Run some of the CI tests in your local workspace.

See [OpenAPI Tools wiki](https://github.com/OpenAPITools/openapi-generator/wiki/Integration-Tests) for more information about the integration tests.

### Tips

- Smaller changes are easier to review

- [Optional] For bug fixes, provide a OpenAPI Spec to repeat the issue so that the reviewer can use it to confirm the fix

- Add test case(s) to cover the change

- Document the fix in the code to make the code more readable

- Make sure test cases passed after the change (one way is to leverage https://travis-ci.org/ to run the CI tests)

- File a PR with meaningful title, description and commit messages

- Make sure the option "Allow edits from maintainers" in the PR is selected so that the maintainers can update your PRs with minor fixes, if needed.

- Recommended git settings

- `git config core.autocrlf input` to tell Git convert CRLF to LF on commit but not the other way around

- To close an issue (e.g. issue 1542) automatically after a PR is merged, use keywords "fix", "close", "resolve" in the PR description, e.g. `fix #1542`. (Ref: [closing issues using keywords](https://help.github.com/articles/closing-issues-using-keywords/))