Contribution to .NET Runtime

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

You can contribute to .NET Runtime with issues and PRs. Simply filing issues for problems you encounter is a great way to contribute. Contributing implementations is greatly appreciated.

## Contribution "Bar"

Project maintainers will merge changes that improve the product significantly and broadly and that align with the [.NET Roadmap](https://github.com/dotnet/core/blob/master/roadmap.md).

Maintainers will not merge changes that have narrowly-defined benefits, due to compatibility risk. The .NET Core codebase is used by several Microsoft products (for example, ASP.NET Core, .NET Framework 4.x, Windows Universal Apps) to enable execution of managed code. Other companies are building products on top of .NET, too. We may revert changes if they are found to be breaking.

Contributions must also satisfy the other published guidelines defined in this document.

## DOs and DON'Ts

Please do:

\* \*\*DO\*\* follow our [coding style](docs/coding-guidelines/coding-style.md) (C# code-specific)

\* \*\*DO\*\* give priority to the current style of the project or file you're changing even if it diverges from the general guidelines.

\* \*\*DO\*\* include tests when adding new features. When fixing bugs, start with

adding a test that highlights how the current behavior is broken.

\* \*\*DO\*\* keep the discussions focused. When a new or related topic comes up

it's often better to create new issue than to side track the discussion.

\* \*\*DO\*\* blog and tweet (or whatever) about your contributions, frequently!

Please do not:

\* \*\*DON'T\*\* make PRs for style changes.

\* \*\*DON'T\*\* surprise us with big pull requests. Instead, file an issue and start

a discussion so we can agree on a direction before you invest a large amount

of time.

\* \*\*DON'T\*\* commit code that you didn't write. If you find code that you think is a good fit to add to .NET Core, file an issue and start a discussion before proceeding.

\* \*\*DON'T\*\* submit PRs that alter licensing related files or headers. If you believe there's a problem with them, file an issue and we'll be happy to discuss it.

\* \*\*DON'T\*\* add API additions without filing an issue and discussing with us first. See [API Review Process](docs/project/api-review-process.md).

## Breaking Changes

Contributions must maintain [API signature](docs/coding-guidelines/breaking-changes.md#bucket-1-public-contract) and behavioral compatibility. Contributions that include [breaking changes](docs/coding-guidelines/breaking-changes.md) will be rejected. Please file an issue to discuss your idea or change if you believe that it may affect managed code compatibility.

## Suggested Workflow

We use and recommend the following workflow:

1. Create an issue for your work.

- You can skip this step for trivial changes.

- Reuse an existing issue on the topic, if there is one.

- Get agreement from the team and the community that your proposed change is a good one.

- If your change adds a new API, follow the [API Review Process](docs/project/api-review-process.md).

- Clearly state that you are going to take on implementing it, if that's the case. You can request that the issue be assigned to you. Note: The issue filer and the implementer don't have to be the same person.

2. Create a personal fork of the repository on GitHub (if you don't already have one).

3. In your fork, create a branch off of master (`git checkout -b mybranch`).

- Name the branch so that it clearly communicates your intentions, such as issue-123 or githubhandle-issue.

- Branches are useful since they isolate your changes from incoming changes from upstream. They also enable you to create multiple PRs from the same fork.

4. Make and commit your changes to your branch.

- [Workflow Instructions](docs/workflow/README.md) explains how to build and test.

- Please follow our [Commit Messages](#commit-messages) guidance.

5. Add new tests corresponding to your change, if applicable.

6. Build the repository with your changes.

- Make sure that the builds are clean.

- Make sure that the tests are all passing, including your new tests.

7. Create a pull request (PR) against the dotnet/runtime repository's \*\*master\*\* branch.

- State in the description what issue or improvement your change is addressing.

- Check if all the Continuous Integration checks are passing.

8. Wait for feedback or approval of your changes from the [area owners](docs/area-owners.md).

- Details about the pull request [review procedure](docs/pr-guide.md).

9. When area owners have signed off, and all checks are green, your PR will be merged.

- The next official build will automatically include your change.

- You can delete the branch you used for making the change.

Note: It is OK for your PR to include a large number of commits. Once your change is accepted, you will be asked to squash your commits into one or some appropriately small number of commits before your PR is merged.

Note: It is OK to create your PR as "[WIP]" on the upstream repo before the implementation is done. This can be useful if you'd like to start the feedback process concurrent with your implementation. State that this is the case in the initial PR comment.

## Up for Grabs

The team marks the most straightforward issues as [up for grabs](https://github.com/dotnet/runtime/labels/up-for-grabs). This set of issues is the place to start if you are interested in contributing but new to the codebase.

## Commit Messages

Please format commit messages as follows (based on [A Note About Git Commit Messages](http://tbaggery.com/2008/04/19/a-note-about-git-commit-messages.html)):

```

Summarize change in 50 characters or less

Provide more detail after the first line. Leave one blank line below the

summary and wrap all lines at 72 characters or less.

If the change fixes an issue, leave another blank line after the final

paragraph and indicate which issue is fixed in the specific format

below.

Fix #42

```

Also do your best to factor commits appropriately, not too large with unrelated things in the same commit, and not too small with the same small change applied N times in N different commits.

## Contributor License Agreement

You must sign a [.NET Foundation Contribution License Agreement (CLA)](https://cla.dotnetfoundation.org) before your PR will be merged. This is a one-time requirement for projects in the .NET Foundation. You can read more about [Contribution License Agreements (CLA)](http://en.wikipedia.org/wiki/Contributor\_License\_Agreement) on Wikipedia.

The agreement: [net-foundation-contribution-license-agreement.pdf](https://github.com/dotnet/home/blob/master/guidance/net-foundation-contribution-license-agreement.pdf)

You don't have to do this up-front. You can simply clone, fork, and submit your pull-request as usual. When your pull-request is created, it is classified by a CLA bot. If the change is trivial (for example, you just fixed a typo), then the PR is labelled with `cla-not-required`. Otherwise it's classified as `cla-required`. Once you signed a CLA, the current and all future pull-requests will be labelled as `cla-signed`.

## File Headers

The following file header is the used for .NET Core. Please use it for new files.

```

// Licensed to the .NET Foundation under one or more agreements.

// The .NET Foundation licenses this file to you under the MIT license.

```

- See [class.cpp](./src/coreclr/src/vm/class.cpp) for an example of the header in a C++ file.

- See [List.cs](./src/libraries/System.Private.CoreLib/src/System/Collections/Generic/List.cs) for an example of the header in a C# file.

## PR - CI Process

The [dotnet continuous integration](https://dev.azure.com/dnceng/public/) (CI) system will automatically perform the required builds and run tests (including the ones you are expected to run) for PRs. Builds and test runs must be clean.

If the CI build fails for any reason, the PR issue will be updated with a link that can be used to determine the cause of the failure.

## PR Feedback

Microsoft team and community members will provide feedback on your change. Community feedback is highly valued. You will often see the absence of team feedback if the community has already provided good review feedback.

One or more Microsoft team members will review every PR prior to merge. They will often reply with "LGTM, modulo comments". That means that the PR will be merged once the feedback is resolved. "LGTM" == "looks good to me".

There are lots of thoughts and [approaches](https://github.com/antlr/antlr4-cpp/blob/master/CONTRIBUTING.md#emoji) for how to efficiently discuss changes. It is best to be clear and explicit with your feedback. Please be patient with people who might not understand the finer details about your approach to feedback.

## Contributing Ports

We encourage ports of CoreCLR to other platforms. There are multiple ports ongoing at any one time. You may be interested in one of the following ports:

Chips:

- [ARM32](https://github.com/dotnet/runtime/labels/arch-arm32)

- [ARM64](https://github.com/dotnet/runtime/labels/arch-arm64)

- [X86](https://github.com/dotnet/runtime/labels/arch-x86)

Operating System:

- [Linux](https://github.com/dotnet/runtime/labels/os-linux)

- [macOS](https://github.com/dotnet/runtime/labels/os-mac-os-x)

- [Windows Subsystem for Linux](https://github.com/dotnet/runtime/labels/os-windows-wsl)

- [FreeBSD](https://github.com/dotnet/runtime/labels/os-freebsd)

Note: Add links to install instructions for each of these ports.

Ports have a weaker contribution bar, at least initially. A functionally correct implementation is considered an important first goal. Performance, reliability and compatibility are all important concerns after that.

### Copying Files from Other Projects

.NET Core uses some files from other projects, typically where a binary distribution does not exist or would be inconvenient.

The following rules must be followed for PRs that include files from another project:

- The license of the file is [permissive](https://en.wikipedia.org/wiki/Permissive\_free\_software\_licence).

- The license of the file is left in-tact.

- The contribution is correctly attributed in the [3rd party notices](./THIRD-PARTY-NOTICES.TXT) file in the repository, as needed.

See [IdnMapping.cs](./src/libraries/System.Private.CoreLib/src/System/Globalization/IdnMapping.cs) for an example of a file copied from another project and attributed in the [CoreCLR 3rd party notices](./THIRD-PARTY-NOTICES.TXT) file.

### Porting Files from Other Projects

There are many good algorithms implemented in other languages that would benefit the .NET Core project. The rules for porting a Java file to C#, for example, are the same as would be used for copying the same file, as described above.

[Clean-room](https://en.wikipedia.org/wiki/Clean\_room\_design) implementations of existing algorithms that are not permissively licensed will generally not be accepted. If you want to create or nominate such an implementation, please create an issue to discuss the idea.