# Contributing to the PerfView repository

First and foremost, we want to thank you for your willingness to help make PerfView better.

Here we describe a bunch of rules / advice associated with doing so. You may end up

getting frustrated with the process. This guide is our attempt to keep that frustration

as low as possible.

The simple truth is that not all change is good (big surprise), and because of this, we reserve

the right to reject any pull request to this repo. We won’t do this without rationale, and

our rationale is simple at its heart.

1. Complexity is bad. Changes that add complexity need more careful deliberation

to weigh the bad against the good that comes along with it.

2. Consistency is good. Bugs are basically a kind of inconsistency (the program does

not work as a simple understanding would assume). There are, of course, feature additions

that make consistency better, but it is all too easy for new features to NOT be consistent

with existing features.

3. In addition to the benefit of the new behavior, you must also carefully consider

cost to any scenario that is being penalized (e.g. make it slower or more complex,

change its UI, or take away functionality).

And our golden rule

\* \*\*When in doubt, ask us by posting an issue or suggestion\*\*

The key here is that from your point of view the new feature you are adding looks like unalloyed

good. But it is very likely that it increases complexity (point 1), may affect consistency (point 2)

and may make other scenarios slower (e.g. startup, etc.) (point 3). As the keepers of this repo we

have the responsibility to weigh these other factors, and we may decide that the bad outweighs

the good.

A rejected pull request is a failure for the repo as a whole because it means that multiple people

spent time on things that ultimately did not benefit the repo. We want to avoid that. There

is a simple heuristic that helps:

\* \*\*The bigger the change, the more 'pre-vetting' you need\*\*

Thus what we DON'T what you to do is over the course of time build up a rather massive change

and then as some point decide to submit it as a pull request. The likelihood of this landing

in a good place is next to nil.

Small bug fixes / features that do not add interesting complexity are easy, just do them and

submit the pull request. Bigger bug fixes should be vetted by asking first. Code

reorganization is particularly tricky. By (1) and (2) we like this assuming it lowers overall

complexity and improves consistency, but it is very disruptive and ideally is done in a series

of small steps. Thus planning is needed, and you should talk with us by posting the issue.

In general all features need pre-vetting. The rule here is simple. Don't do any work unless

you:

1. Are willing to throw it away (e.g. it was more effort to get it vetted) OR

2. You have vetted it with us by creating an issue, and got positive confirmation from the

discussion that you are on the right track. If it is a big change, you should stay updated with ongoing

discussion on the issue to insure that you stay on track.

Performance improvements are often a point of contention. Improvements that make the code

smaller / simpler are great, but often this is not the case. If you are adding complexity as

part of your improvement (e.g. adding a cache), again, you have to follow the rule above

and get it pre-vetted, or be willing to abandon the change. For performance changes in

general we will probably ask you to take measurements to quantify exactly how much improvement

there was. There is more work than just modifying the code.

## Coding Standards

See [PerfView Coding Standards](documentation/CodingStandards.md).

## Testing and Contributing tests

There are a number of \*.Test projects that are unit tests that need to be run before checking in.

You can run these tests in Visual Studio by selecting the Test -> Run -> All Tests menu entry.

These tests need to be run on the DEBUG configuration (that is with asserts) for them to

have really good effectiveness (the code has lots of asserts).

The tests should run in less than 1 minute total.