Contributing PHP changes

How a PHP change reaches the Exponential kernel - the contributor license agreement, fork and branch, the checks to run before a pull request, commit messages, tests, the file manifest, review rounds, squashing and...

Source file: `doc/eng-us/help/php-developers/contributing-php-changes.md` Last changed 2026-11-04
https://doc.exponential.earth/learninminutes/exphelp/viewer/help/php-developers/contributing-php-changes.md

This page is for PHP developers who want a fix or a feature in the Exponential kernel itself, the repository `se7enxweb/exponential`. The binding text is `CONTRIBUTING.md` at the root of the repository; this page puts its steps in the order a PHP developer meets them and links to the checks.

Contents

1. Kernel or extension?
2. Before your first pull request: the CLA
3. Fork, clone, branch
4. Write the change
5. Check it
6. Commit it
7. The pull request and the review
8. After the merge

Kernel or extension?

Contribute to the kernel	Keep in an extension
a bug fix in kernel or library code	a feature only your sites need
a PHP compatibility fix	a module, datatype or operator of your own
a new extension point the kernel lacks	code that uses existing extension points
documentation of existing behavior	

Your own extensions are not affected by the kernel's contribution terms. Publishing an extension of your own is described in Contribute extensions back and the extension developer book, part 5.

Before your first pull request: the CLA

7x and the Exponential Foundation require a signed Contributor Licensing Agreement for every code contribution to the core kernel, before it can be merged. Read the agreement and the summary in `CONTRIBUTING.md` ("Copyright assignment (CLA)") before you sign: by signing and submitting a contribution to the core kernel, you assign the copyright in that contribution to the Exponential Foundation and 7x. You sign once for all your future contributions; a pull request

without a signed CLA on file is reviewed but not merged until it is received.

Fork, clone, branch

Shell

```
git clone https://github.com/<your-account>/exponential.git
cd exponential
composer install
make devtools && make hooks
git checkout -b fix/acme-short-description
```

Set up an installation to test on (Setting up a PHP development checkout).

Write the change

- Follow the code style of `phpcs.xml.dist` and the files around your change (The coding standard).
- Keep PHP 8.0 compatibility (Writing code that runs on PHP 8.0).
- Name new code with the `exp` prefix; never rename existing `eZ*` identifiers (Class naming and the exp prefix).
- Admin templates and stylesheets follow Styling the administration: no `<style>` blocks or `style=""` attributes, shared components and tokens.
- Add a test that fails without your change and passes with it (PHPUnit without a database).
- Update the documentation the change affects.

Check it

Shell

```
make check                                  # syntax, manifest, code style, PHPStan
make lint80                                 # PHP 8.0 syntax of changed files (needs a PHP 8.0 b
inary)
php vendor/bin/phpunit <your test files>
make manifest-fix                           # refresh share/filelist.md5 for added and changed f
iles
```

CI and reproducing a red check lists what CI runs and how to reproduce each step.

Commit it

The subject starts with what the change does, and the message says why:

```
Fixed: The redirect keeps the query string on the own host

The query string was dropped when the target host was the current one, so
paged lists lost their page after a save.
```

Prefixes: `Added:` , `Updated:` , `Fixed:` , `Removed:` , `Renamed:` . The commit-msg hook checks them (Commit messages). Commit `share/filelist.md5` with the change.

The pull request and the review

1. Push your branch to your fork and open a pull request to `main`; tick the CLA checkbox of the template.
2. **Expect many review rounds.** The project reviews more than usual before it accepts a change; each round makes the change easier to accept. Answer every comment in the pull request, with a change or a reason. A change to a negative feature, behavior that is deliberate or open to interpretation and that other sites depend on, needs a clear justification and goes in more easily as an option than as a new default.
3. **Keep it small and focused**: a fix and a refactoring are two pull requests.
4. **Squash and rebase while it is open.** Fold the fixups of each round into logical commits and rebase onto the current `main`; force-push only your pull request branch (`git push --force-with-lease`).
5. A pull request is tested as merged into `main`; when `main` moves, merge or rebase and push again.

After the merge

`main`'s history is never rewritten after a merge: a later fix comes as a new pull request. Releases and tags are made by the project; a published version never changes.

Related pages

- Develop with Git
- Contributing back with pull requests
- Source control and releases standard
- Free documentation: this page is free to copy, translate and improve under the GNU FDL 1.3.

> **Become an Exponential Partner**
>
> Partners share their information, provide funding, contribute back and develop the products with us, and that is how Exponential grows. A partner website and dashboard are planned for mid-2027: please become a partner today to speed up our growth rate and land new features faster.
>
> Partners: start here | How to apply

Exponential Copyright © 1998-2026 Exponential Foundation and 7x
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".
This documentation is free documentation, licensed under the GNU Free Documentation License 1.3 or later:
https://www.gnu.org/licenses/fdl-1.3.html.