Handling technical debt

How technical debt is found, recorded and reduced in Exponential - the known gaps sections of the standards, the PHPStan baseline, checks that fail only on new problems, deprecations without removal dates, plain-file...

Source file: `doc/eng-us/help/software-engineers/technical-debt.md` Last changed 2026-11-04
https://doc.exponential.earth/learninminutes/exphelp/viewer/help/software-engineers/technical-debt.md

This page is for engineers who work in a code base with a long history: the code base was developed from 1999, first under the name eZ Publish, a product its creator discontinued after almost 18 years, and Exponential continues it. Debt is expected. What matters is that it is visible, that it does not grow, and that it is paid down without breaking installations.

Contents

1. Where debt is recorded
2. Debt that cannot grow
3. Examples on record
4. Paying it down
5. What not to do

Where debt is recorded

Record	What it holds
Known gaps sections of the standards in `doc/eng-us/standards/`	what the project does not meet yet, with evidence (Kernel architecture, Known exceptions and debt)
`phpstan-baseline.neon`	the PHPStan findings the code had when the check was introduced
`@deprecated` tags in `kernel/` and `lib/`	APIs with a replacement, still working
Upgrade notes in `doc/eng-us/bc/6.0/`	changes that needed action, and their reasons
The issue tracker	individual problems

Debt that cannot grow

The project's checks are built so that old problems are tolerated and new ones are refused:

- PHPStan fails only on findings missing from the baseline.
- Code style fails only when a change adds violations to a file.
- The design tests of the administration have an empty allowlist; an entry may not be added to make a change pass.

A change may leave an old problem in place; it must not add one. See Static analysis and code style.

Examples on record

From the standards pages:

- `lib/` depends on kernel classes in several files, so it is "lower level than the kernel", not independent.
- A few view classes still read `$_POST` or `$_GET` directly.
- `ezpEvent` is documented as private API, yet extensions rely on it.
- Plain-file views, cronjob parts and commands still run beside the class-based form.
- INI switches mix `enabled`/`disabled` and `true`/`false`.
- Installations upgraded by hand with the SQL update files have no record of what ran; `exp:upgrade` and the migrations (6.0.15) record it.
- Some test files are in no suite and never run.
- `@deprecated` APIs inherited from the old code base have no removal date.

Paying it down

1. **Small and separate.** Raise the PHPStan level or shrink the baseline in small separate changes; one refactoring per pull request, never mixed with a fix.
2. **Keep entry points.** When code moved into classes in 6.0.15, every view, script and cronjob part kept its path as a one-line call, so includes, cron entries and `--help` output did not change. Do the same.
3. **Deprecate, do not remove.** Mark a replaced API `@deprecated` with its replacement and keep it working; announce a removal in the changelog and an upgrade note (Backward compatibility policy).
4. **Test before you move.** Add a test that pins the current behavior, then change the code.
5. **Update the record.** When a known gap closes, remove it from the standard's list in the same change.

What not to do

- Do not regenerate the baseline to make a new finding pass.
- Do not rename existing `eZ*` identifiers, tables or settings to modernize them.
- Do not reformat whole files in a functional change: reviewers cannot see the change.
- Do not add type declarations to existing kernel signatures: extensions that override them break.

Related pages

- Kernel, lib and extensions: the layering
- Why most changes are small
- Inventory an old extension
- 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.