The engine archive (phar) for PHP developers

What the optional engine archive dist/engine.phar means for PHP code - which directories it holds, how it is switched on, why code must never compute the installation root from __DIR__, how rebuilds work and refuse...

Source file: `doc/eng-us/help/php-developers/the-engine-phar-for-php-developers.md` Last changed 2026-11-04
https://doc.exponential.earth/exphelp/viewer/help/php-developers/the-engine-phar-for-php-developers.md

This page is for PHP developers whose code may run with the engine archive switched on. Exponential can load `kernel/`, `lib/` and `autoload/` from one archive, `dist/engine.phar`, instead of from a thousand separate files. It is off by default, it is not a performance feature (measured, it makes no difference), and it is not source protection. But it changes one thing PHP code must respect: paths. The full reference is Engine archive (phar) support.

Contents

1. What is in the archive
2. The rule for your code: never derive the root from a kernel file
3. Switching it on and off
4. Builds and rebuilds
5. The phar stream wrapper
6. Recognizing it in a stack trace

What is in the archive

From the archive	From disk, always
`kernel/`, `lib/`, `autoload/`	`extension/`, `design/`, `settings/`, `share/`, `bin/`, `var/`, `vendor/`

Only the autoloader behaves differently: a kernel or library class is read through `phar://.../engine.phar/...`. Your extension's classes are always read from disk.

The rule for your code: never derive the root from a kernel file

Inside the archive, `__DIR__` and `__FILE__` of a kernel file name a path **inside the archive**, so code that computes the installation root as `dirname( __DIR__, 3 )` finds nothing. `autoload.php`, which is always on disk, publishes the root once:

PHP

```php
// the installation root, an absolute path, whatever the engine
$root = defined( 'EXP_ROOT_DIR' ) ? EXP_ROOT_DIR : eZSys::rootDir();
$file = $root . '/var/acmereport/export.csv';
```

- Use `EXP_ROOT_DIR` (defined by `autoload.php`) or `eZSys::rootDir()`.
- Use `eZSys::varDirectory()`, `eZSys::cacheDirectory()` and `eZSys::storageDirectory()` for the site's directories.
- Use paths relative to the installation root for settings and designs, as the kernel does; the current directory of every entry point is the root.

Your extension's own files are on disk, so `__DIR__` inside an extension class still works. The rule matters when you change kernel or library code, and for helpers that may move into the kernel later.

Switching it on and off

INI

```ini
# settings/velocity.ini, or an override
[ServerSettings]
EnginePhar=enabled          ; disabled (default), enabled, or a path to an archive
```

or the environment variable `EXP_ENGINE_PHAR=/path/to/dist/engine.phar` for whatever serves the site. Velocity exports the variable to its workers. An archive that is missing stops the server from starting rather than silently falling back. Clear the caches when you switch, in either direction.

Builds and rebuilds

Shell

```shell
php bin/php/console exp:phar build      # write dist/engine.phar (only when something changed)
php bin/php/console exp:phar check      # current or not, which files differ; exit 1 when stale
php bin/php/console exp:phar info       # path, size, version, whether it is in use
php bin/php/console exp:phar clean      # remove it
```

- `dist/engine.phar.index.json` records every packaged file with size, time and SHA-1. A build, and every `exp:velocity start`, `restart` and `graceful`, rebuilds only when a file of `kernel/`, `lib/` or `autoload/` was added, removed or changed, or PHP's minor version changed.
- **A file that does not parse keeps the old archive.** Every PHP file is parsed first; a broken one is named and nothing is written. A restart then refuses before stopping the server. Run `php -l` on what you changed.
- The archive is stamped with the release, the commit and `-dirty` for an uncommitted tree; `exp:phar info` and **Setup > System information** (block *Engine*) show it.

So with the archive on, **a kernel edit is live only after a rebuild**, which `exp:velocity deploy` and a restart do for you.

The phar stream wrapper

`autoload.php` unregisters PHP's `phar://` stream wrapper on every request, unless the runtime itself is read through it (the engine archive, or a tool such as `phpunit.phar` as the entry point). The reason is security: a file that is both a valid image and a valid phar can be uploaded, and with the wrapper registered `include( 'phar://<upload>.jpg/x.php' )` would run the PHP

inside it. Do not register the wrapper again, and do not use `phar://` paths in your own code.

Recognizing it in a stack trace

A stack trace or an error message with `phar://.../dist/engine.phar/kernel/...` means the archive is in use. If an edit to a kernel file "does nothing", check `exp:phar check` and **Setup > System information** first.

Related pages

- Engine archive (phar) support
- Velocity persistent workers for PHP developers
- Velocity engines
- 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.