Plain CLI scripts with eZScript and ezexec

The classic way to write a command line script with eZScript and eZCLI, which still works and which you will meet in older extensions - instance(), startup(), getOptions(), initialize(), shutdown() - plus running...

Source file: `doc/eng-us/help/php-developers/legacy-cli-scripts-with-ezscript.md` Last changed 2026-11-04
https://doc.exponential.earth/learninminutes/exphelp/viewer/help/php-developers/legacy-cli-scripts-with-ezscript.md

This page is for PHP developers who maintain command line scripts written as plain files, or who need to run a piece of PHP with the kernel booted. New commands are written as classes (Write a console command); the plain form still runs exactly as before, and every class-based command uses the same `eZScript` underneath, so knowing it helps either way.

Contents

1. The life cycle
2. A plain script
3. The settings of eZScript::instance()
4. eZCLI output
5. Running code with the kernel: ezexec.php
6. Moving a plain script to a class

The life cycle

Step	Call	Does
1	`eZScript::instance( $settings )`	creates the script object
2	`$script->startup()`	checks that it runs on the command line, sets error reporting and the debug handling
3	`$script->getOptions( $config, $argumentConfig, $help )`	parses the command line; prints `--help` and exits; applies `-s`, `-l`, `-d`, `-v`, `-q`
4	`$script->initialize()`	loads the siteaccess, the extensions, the session and the database
5	your work	
6	`$script->shutdown( $exitCode, $exitText )`	writes the debug report if asked, and exits

Do the option parsing before `initialize()`, so `--help` answers without a database.

A plain script

PHP

```php
#!/usr/bin/env php
<?php
/**
 * @description Counts the items of acmereport
 */
require_once 'autoload.php';

$cli = eZCLI::instance();
$script = eZScript::instance( array( 'description' => "Counts the items of acmereport",
                                     'use-session' => false, 'use-modules' => false, 'use-extensi
ons' => true ) );
$script->startup();
$options = $script->getOptions( '[status:]', '', array( 'status' => 'Only items with this status'
 ) );
$script->initialize();

$conds = $options['status'] ? array( 'status' => (string)$options['status'] ) : null;
$cli->output( 'Items: ' . eZPersistentObject::count( acmeReportItem::definition(), $conds ) );

$script->shutdown( 0 );
```

Run it from the installation root: `php extension/acmereport/bin/php/count.php --status=open`, or through the console as `ext:acmereport:count`.

The settings of eZScript::instance()

Key	Meaning
`description`	the text of `--help`
`use-session`	start a session (most scripts do not need one)
`use-modules`	load the module system (needed to run views or check policies of modules)
`use-extensions`	load the active extensions and their settings (almost always `true`)
`site-access`	a fixed siteaccess instead of the default

eZCLI output

Call	Writes
`$cli->output( $text )`	a line to standard output; nothing with `-q`
`$cli->notice( $text )`, `warning( $text )`, `error( $text )`	marked lines
`$cli->stylize( 'emphasize', $text )`	text with a terminal style; plain when colors are off

Running code with the kernel: ezexec.php

To run a PHP file once with the kernel booted (settings, extensions, database), for a diagnosis or a one-time data fix, pass it to `bin/php/ezexec.php`:

Shell

```
php bin/php/ezexec.php path/to/your-script.php
php bin/php/ezexec.php tests/bin/bootsmoke.php      # how CI boots the kernel on PHP 8.0
```

The file runs with the kernel's classes and settings available; `eZDB::instance()` connects to the site's database. Keep such files out of the web root's reach and out of your extension's releases, and give them a name that says what they do. Anything you run twice belongs in a proper command.

Moving a plain script to a class

1. Create `extension/<ext>/classes/runnable/commands/<name>.php` with a class extending `Exponential\Runnable\Command` in `Exponential\Command\Extension\<Ext>`.
2. Move the code into `run()`; replace `eZScript::instance()` with `$this->script()`, the three start steps with `$this->startup()`, `$cli->output()` with `$this->output()`, `$script->shutdown()` with `$this->shutdown()`.
3. Write class names fully qualified (`\eZCLI`), because the class is in a namespace.
4. Reduce the old file to `require_once 'autoload.php';` and one `main( __FILE__ )` call.
5. Compare `--help` and the exit codes before and after; the kernel proved its own moves this way (Commands, cronjob parts and module views as classes).

Related pages

- eZScript and eZCLI reference
- CLI scripts with eZScript (application creators)
- CLI scripts and console commands
- 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.