Command line: the console and eZScript

The Exponential counterpart of bin/console, artisan, manage.py, rails commands and Spring Boot runners - bin/php/console with its exp:, ext:, shell: and cron: namespaces, writing a command as an...

Source file: `doc/eng-us/help/web-framework-developers/cli-and-console-commands.md` Last changed 2026-11-04
https://doc.exponential.earth/learninminutes/exphelp/viewer/help/web-framework-developers/cli-and-console-commands.md

This page is for developers who live in `bin/console`, `php artisan`, `manage.py` or `bin/rails`. Exponential has one entry point for every command, `bin/php/console` (installations usually add a `./console` shortcut in the root). It finds commands at run time from their file location, so an extension's command needs no registration (Exponential Console, Console commands).

Contents

1. Finding commands
2. Command names
3. Writing a command
4. Siteaccess, user and root
5. Commands you will use daily
6. Compared with other frameworks

Finding commands

Shell

```
php bin/php/console                   # every command
php bin/php/console list ext:expdemo  # one namespace
php bin/php/console --version
```

Command names

Namespace	Runs	Example
`exp:<name>`	`bin/php/<name>.php`	`exp:ezcache`
`shell:<name>`	`bin/shell/<name>.sh`	
`bin:<name>`	`bin/<name>.sh` or `.php`	
`ext:<ext>:<name>`	`extension/<ext>/bin/php/<name>.php` or `bin/<name>.sh`	`ext:expqueue:status`
`cron:<part>`	a cronjob part	`cron:frequent`

A script can declare extra names with an `@alias` tag in its docblock (`bin/php/exphelp.php` declares `@alias exp:help`), and a `@description` tag describes it in `list`.

Writing a command

New commands are classes extending `Exponential\Runnable\Command`, called by a one-line script (CLI scripts with eZScript):

PHP

```php
<?php
// extension/expdemo/classes/runnable/commands/report.php
namespace Exponential\Command\Extension\Expdemo;

class Report extends \Exponential\Runnable\Command
{
    public function run()
    {
        $this->script( array( 'description' => "Writes the monthly report",
                              'use-session' => false, 'use-modules' => true, 'use-extensions' =>
true ) );
        $options = $this->startup( '[month:][dry-run]', '', array(
            'month'   => 'The month, YYYY-MM',
            'dry-run' => 'Show what would be written, write nothing' ) );
        $this->output( \expDemoReportService::build( $options['month'], !empty( $options['dry-run
'] ) ) );
        $this->shutdown( 0 );
    }
}
```

PHP

```php
#!/usr/bin/env php
<?php
// extension/expdemo/bin/php/report.php
require_once 'autoload.php';
\Exponential\Command\Extension\Expdemo\Report::main( __FILE__ );
```

Run it: `php bin/php/console ext:expdemo:report --month=2026-10 --dry-run`. The classic form, used by many scripts, calls `eZScript::instance()`, `startup()`, `getOptions()`, `initialize()` and `shutdown()` directly; output goes through `eZCLI` (`output()`, `notice()`, `warning()`, `error()`). Option strings use `[name:]` for a value and `[name]` for a flag; `--help`, `-s/--siteaccess`, `--allow-root-user` and the debug options are added for you.

Siteaccess, user and root

- `-s <siteaccess>` selects the settings; extensions active only in that siteaccess need it.
- The user is anonymous. A command that writes content signs in a dedicated user (`eZUser::setCurrentlyLoggedInUser( $user, $user->id() )`), never the shared administrator.
- Scripts refuse to run as root unless `--allow-root-user` is given; run them as the web server's user, because files a root process writes into `var/` break the site later.
- The exit code is returned unchanged: `shutdown( 0 )` for success, anything else for failure.
- Offer `--dry-run` for every command that writes or deletes, and never accept passwords as options.

Commands you will use daily

Command	Does
`exp:ezpgenerateautoloads -e`	regenerate extension autoloads after adding classes
`exp:ezcache --clear-tag=ini`	clear caches by tag or id
`exp:cache status`	every cache layer, PASS or FAIL
`exp:ini get/set/where`	read and change settings
`exp:velocity deploy`	autoloads, caches, PHP reload, Velocity restart, in order
`exp:install`	a new installation in one command (Installing in one command)
`exp:help check`, `exp:help search`	the help's link check and search
`exp:maintenance on/off`	maintenance mode
`cron:<part>`	run a cronjob part

Compared with other frameworks

Framework	Concept	Exponential
Symfony	commands registered as services, `#[AsCommand]`	files found by location, `Exponential\Runnable\Command`
Laravel	Artisan commands, `$signature`	the option string of `startup()` / `getOptions()`
Django	management commands in `management/commands/`	scripts in `extension/<ext>/bin/php/`
Rails	rake tasks, `bin/rails` commands	the same
Spring	`CommandLineRunner`	a command class

Related pages

- Commands, cronjob parts and module views as classes
- Command pattern
- Scheduled tasks: cronjobs
- 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.