Debugging with eZDebug

The kernel's debug API for PHP developers - eZDebug::writeNotice, writeWarning, writeError and writeDebug with __METHOD__ labels, conditional debug with eZDebugSetting and debug.ini, timing points and accumulators, who...

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

This page is for PHP developers who need to see what their code does inside a request. `eZDebug` (`lib/ezutils/classes/ezdebug.php`) collects messages, timing points and accumulators, writes warnings and errors to the logs, and shows everything in the debug bar when debug output is on. The bar itself is described on The debug bar for PHP developers.

Contents

1. Writing messages
2. Conditional debug messages
3. Timing points and accumulators
4. Turning debug output on
5. Who gets debug output
6. Debugging on the command line
7. Other tools
8. A working order

Writing messages

PHP

```php
eZDebug::writeNotice( 'Imported ' . $count . ' items', __METHOD__ );
eZDebug::writeWarning( 'Item ' . (int)$id . ' had no title; used the id', __METHOD__ );
eZDebug::writeError( 'The feed could not be read: ' . $message, __METHOD__ );
eZDebug::writeDebug( $details, __METHOD__ );
```

Method	Level	Shown and logged
`writeError()`	error	in the debug output; written to `var/log/error.log` always (the shipped `AlwaysLog[]=error`)
`writeWarning()`	warning	in the debug output; written to `var/log/warning.log` when the level is logged
`writeNotice()`	notice	in the debug output; logged only when listed in `AlwaysLog[]` (it fills logs quickly)
`writeDebug()`	debug	in the debug output; logged only when listed in `AlwaysLog[]`

`site.ini [DebugSettings] AlwaysLog[]` lists the levels that are logged even when debug output is off; the shipped value is `error`. PHP deprecations go to `var/log/deprecated.log` (`DeprecatedLog=enabled`), each distinct message once per request. Pass `__METHOD__` as the label: it names the class and method in every line, so a reader of the log finds the code. Never write passwords, tokens or personal data into a message.

Conditional debug messages

Verbose diagnostics that you want only when you ask for them use a **debug condition**:

PHP

```php
eZDebugSetting::writeDebug( 'extension-acmereport', $details, __METHOD__ );
```

The message is written only when the condition is enabled in `debug.ini`:

INI

```ini
# settings/override/debug.ini.append.php (development)
[GeneralCondition]
extension-acmereport=enabled
```

The kernel uses conditions such as `kernel-content-view`, `kernel-urltranslator` and `workflow-event`; `ezformtoken` uses `ezformtoken`. The debug bar finds every condition of `[GeneralCondition]` and shows it as a switch.

Timing points and accumulators

PHP

```php
eZDebug::addTimingPoint( 'acmereport: before import' );

eZDebug::createAccumulatorGroup( 'acmereport_total', 'Acme report' );
eZDebug::accumulatorStart( 'acmereport_fetch', 'acmereport_total', 'Fetch the feed' );
$feed = acmeReportFeed::fetch( $url );
eZDebug::accumulatorStop( 'acmereport_fetch' );
```

Timing points mark moments of the request; accumulators add up the time of a piece of code across all its calls (and count them). Both appear in the bar's **Timing** tab. The database drivers record their statements as accumulators, which is where the bar's SQL figures come from.

Turning debug output on

On a development machine only (better in a `site.ini.dev.append.php`, see Setting up a PHP development checkout):

INI

```ini
[DebugSettings]
DebugOutput=enabled

[TemplateSettings]
ShowUsedTemplates=enabled
ShowTemplatePathComments=enabled

[DatabaseSettings]
SQLOutput=enabled
```

`ShowTemplatePathComments` writes the template path as an HTML comment around each template's output (Template path comments). Never enable debug output for visitors of a production site: it shows internals.

Who gets debug output

Setting	Meaning
`[DebugSettings] DebugByIP=enabled` with `DebugIPList[]`	only these addresses or networks (IPv4 and IPv6, CIDR, with an optional label and expiry)
`DebugByUser=enabled` with `DebugUserIDList[]`	only these users
`DebugIPList[]=commandline`	the command line too

INI

```
DebugIPList[]=203.0.113.7/32 ; Laptop ; expires=2026-10-02T15:00
```

(Debug bar: the IP list).

Debugging on the command line

Every `eZScript` command takes `-d` (or `--debug`) with the items to show, for example `-d all` or `-d error,warning,timing`:

Shell

```
php bin/php/console ext:acmereport:report --month=2026-09 -d all
```

Other tools

- **Xdebug**: works as usual on PHP-FPM and the command line; `tests/xdebug.ini` is the configuration the PHPUnit children use. Keep it off when you measure performance.
- **The logs**: everything that went wrong without debug output is in `var/log/` (Logs, the log viewer and request logs).
- **An error reference**: error pages show a reference that `php bin/php/explogs.php --reference=<reference>` finds in the logs (Finding an error by its reference).

A working order

1. Reproduce on a development copy with debug output on.
2. Read the **Messages** tab for errors and warnings from your files.
3. Check the **Templates** tab and the template path comments: is your code even the one that runs?
4. Add `eZDebug::writeDebug()` or a condition around the suspect code.
5. Under Velocity, compare with PHP's built-in server (`--engine=php`): if the bug disappears, look for state kept between requests (Static state traps).
6. Write a test that fails, fix, and keep the test.

Related pages

- eZDebug and logging reference
- Debugging and logs (application creators)
- Debug output improvements
- 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.