PHP 8 migration notes for your code

What to change in older PHP code so it runs cleanly on PHP 8.0 to 8.5 (PHP 7.4 usable without support) under Exponential 6 - dynamic properties, count() on non-arrays, null passed to string functions, SPL return types...

Source file: `doc/eng-us/help/php-developers/php-8-migration-notes-for-your-code.md` Last changed 2026-11-04
https://doc.exponential.earth/learninminutes/exphelp/viewer/help/php-developers/php-8-migration-notes-for-your-code.md

This page is for PHP developers who bring older code, often an extension written for PHP 5 or 7, up to PHP 8 for Exponential 6. The kernel went through the same work between December 2023 and April 2026 (PHP 8 support); this page turns its lessons into a list you can work through.

Contents

1. How to find the problems
2. Declare the properties you use
3. Test before you count
4. Do not pass null to string functions
5. Mark methods of SPL interfaces
6. Nullable parameters
7. Removed functions and constants
8. Signatures that must match the kernel
9. A working order

How to find the problems

- **Debug output on a development copy.** Enable `[DebugSettings] DebugOutput=enabled` and use the pages of your extension; look for "Deprecated" notices that name your files.
- **The logs.** `var/log/warning.log` and `var/log/error.log` collect what the debug output shows (Logs, the log viewer and request logs).
- **Static checks.** `.devtools/vendor/bin/phpcs --standard=phpcs.xml.dist extension/<ext>/` reports PHPCompatibility findings (removed and deprecated functions), and PHPStan finds undefined methods and wrong argument counts (Static analysis with PHPStan).
- **Syntax on the oldest and newest PHP.** `php -l` with 8.0 and with 8.5.

Declare the properties you use

PHP 8.2 deprecates creating a property that the class does not declare. Old extensions do this constantly, often in classes extending `eZPersistentObject` or `eZDataType`:

PHP

```php
class acmeReport extends eZPersistentObject
{
    /** declared, so PHP 8.2 and later do not warn */
    public $ID;
    public $Title;
}
```

`eZPersistentObject` subclasses no longer cause warnings in the debug output since 6.0.4, but declaring the properties is still the clean fix, and it documents the class (Dynamic properties).

Test before you count

`count()` throws a `TypeError` on `null` and scalars since PHP 8.0:

PHP

```php
// before
if ( count( $list ) > 0 ) { ... }

// after
if ( is_array( $list ) && count( $list ) > 0 ) { ... }
// or, for arrays and Countable objects
if ( is_countable( $list ) && count( $list ) > 0 ) { ... }
```

The same applies to `foreach` over a value that may be `null`: guard it or default it (`$list ?? array()`).

Do not pass null to string functions

PHP 8.1 deprecates `null` for non-nullable internal parameters (`strlen( null )`, `trim( null )`, `preg_match( $p, null )`). Values from settings, the database and `$Params` are often `null`:

PHP

```php
$title = trim( (string)( $row['title'] ?? '' ) );
```

The kernel fixed this in `mysqli_real_escape_string()` and `preg_match()` calls in 6.0.11.

Mark methods of SPL interfaces

Methods implementing `Iterator`, `ArrayAccess`, `Countable` or `IteratorAggregate` must declare PHP 8.1's tentative return types or carry the attribute. The kernel style has no return types, so use the attribute:

PHP

```php
#[\ReturnTypeWillChange]
public function current()
{
    return $this->items[$this->position];
}
```

PHPStan cannot baseline a missing tentative return type; that is why `phpstan.neon.dist` excludes `dfs_filter_iterator.php` until it is fixed.

Nullable parameters

PHP 8.4 deprecates a parameter that has a class type and a `null` default without `?` (implicitly nullable). Write the `?` explicitly; nullable class hints are allowed by the coding standard:

PHP

```php
// before
public static function reportDirectory( eZINI $siteINI = null )
// after
public static function reportDirectory( ?eZINI $siteINI = null )
```

See Nullable parameters and types.

Removed functions and constants

Removed	Since	Use
`each()`	8.0	`foreach`, or `key()` and `current()`
`create_function()`	8.0	a closure
`split()`, `ereg()`, `eregi()`, `ereg_replace()`	7.0	`explode()`, `preg_split()`, `preg_match()`, `preg_replace()`
`mysql_*`	7.0	eZDB
`E_STRICT`	deprecated in 8.4	leave it out of error masks
`utf8_decode()`, `utf8_encode()`	deprecated in 8.2	`mb_convert_encoding()`

`phpcs.xml.dist` forbids `each`, `create_function`, `split`, `ereg`, `eregi`, `ereg_replace`, `mysql_query`, `mysql_connect`, `sizeof` and `delete` (The coding standard).

Signatures that must match the kernel

A class that overrides a kernel method must keep its parameters: same count (more with defaults is fine), same by-reference markers. On PHP 8 a mismatch is a fatal error **when the class loads**, so the whole page breaks, not only the method. Compare with the parent in `kernel/classes/` when you upgrade an extension; the boot smoke test finds these (Writing code that runs on PHP 8.0).

A working order

1. Make every file pass `php -l` on 8.0 and 8.5.
2. Load every class (boot smoke test or open every page): fix signatures.
3. Fix `count()`, `null` and property warnings from the debug output, page by page.
4. Run PHPCompatibility and PHPStan on the extension.
5. Raise the `composer.json` PHP constraint to `^8.0` and release (The upgrade checklist).

Related pages

- PHP 8.1 to 8.5 deprecations
- PHP 8 errors that stop old code
- Database code for PHP 8
- 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.