Other backward incompatible changes

- New objects cannot be assigned by reference
- Invalid class, interface and trait names
- ASP and script PHP tags removed
- Calls from incompatible context removed
- ENTITY is now a right associative operator
- Functions cannot have multiple parameters with the same name
- Functions inspecting arguments report the current parameter value
- Switch statements cannot have multiple default blocks
- $HTTP_RAW_POST_DATA removed
- # comments in INI files removed
- JSON extension replaced with JSOND
- Internal function failure on overflow
- Fixes to custom session handler return values
- Sort order of equal elements
- Misplaced break and continue statements
- Constant disallowed as break and continue argument
- Mhash is not an extension anymore
- declare(ticks)

New objects cannot be assigned by reference

The result of the ENTITY statement can no longer be assigned to a variable by reference:

```
<?php
class C {}
$c =& new C;
?>
```

ENTITY

```
Deprecated: Assigning the return value of new by reference is deprecated in /tmp/test.php on line 3
```

ENTITY

```
Parse error: syntax error, unexpected 'new' (T_NEW) in /tmp/test.php on line 3
```

Invalid class, interface and trait names

The following names cannot be used to name classes, interfaces or traits:

- `bool`
- `int`
- `float`
- `string`
- ENTITY
- ENTITY
- ENTITY

Furthermore, the following names should not be used. Although they will not generate an error in PHP 7.0, they are reserved for future use and should be considered deprecated.

- `resource`
- `object`
- `mixed`
- `numeric`

ASP and script PHP tags removed

Support for using ASP and script tags to delimit PHP code has been removed. The affected tags are:

Opening tag	Closing tag

`ENTITY%`	`%ENTITY`
`ENTITY%=`	`%ENTITY`
`ENTITYscript language="php"ENTITY`	`ENTITY/scriptENTITY`

Removed ASP and script tags

Calls from incompatible context removed

Previously deprecated in PHP 5.6, static calls made to a non-static method with an incompatible context will now result in the called method having an undefined `$this` variable and a deprecation warning being issued.

```
<?php
class A {
    public function test() { var_dump($this); }
}

// Note: Does NOT extend A
class B {
    public function callNonStaticMethodOfA() { A::test(); }
}

(new B)->callNonStaticMethodOfA();
?>
```

ENTITY

```
Deprecated: Non-static method A::test() should not be called statically, assuming $this from
incompatible context in /tmp/test.php on line 8
object(B)#1 (0) {
}
```

ENTITY

```
Deprecated: Non-static method A::test() should not be called statically in /tmp/test.php on
line 8
```

```
Notice: Undefined variable: this in /tmp/test.php on line 3
NULL
```

ENTITY is now a right associative operator

The ENTITY construct no longer requires parentheses, and has been changed to a right associative operator with precedence between `print` and `=ENTITY`. This can result in changed behaviour:

```
<?php
echo yield -1;
// Was previously interpreted as
echo (yield) - 1;
// And is now interpreted as
echo yield (-1);

yield $foo or die;
// Was previously interpreted as
yield ($foo or die);
// And is now interpreted as
(yield $foo) or die;
?>
```

Parentheses can be used to disambiguate those cases.

Functions cannot have multiple parameters with the same name

It is no longer possible to define two or more function parameters with the same name. For example, the following function will trigger an `E_COMPILE_ERROR`:

```
<?php
function foo($a, $b, $unused, $unused) {
    //
}
?>
```

Functions inspecting arguments report the *current* parameter value

`func_get_arg`, `func_get_args`, `debug_backtrace` and exception backtraces will no longer report the original value that was passed to a parameter, but will instead provide the current value (which might have been modified).

```
<?php
function foo($x) {
    $x++;
    var_dump(func_get_arg(0));
}
foo(1);?>
```

ENTITY

```
1
```

ENTITY

```
2
```

Switch statements cannot have multiple default blocks

It is no longer possible to define two or more default blocks in a switch statement. For example, the following switch statement will trigger an `E_COMPILE_ERROR`:

```
<?php
switch (1) {
    default:
    break;
```

```
    default:
    break;
}
?>
```

`$HTTP_RAW_POST_DATA` removed

`$HTTP_RAW_POST_DATA` is no longer available. The `php://input` stream should be used instead.

`#` comments in INI files removed

Support for prefixing comments with `#` in INI files has been removed. `;` (semi-colon) should be used instead. This change applies to ENTITY, as well as files handled by `parse_ini_file` and `parse_ini_string`.

JSON extension replaced with JSOND

The JSON extension has been replaced with JSOND, causing three minor BC breaks. Firstly, a number must not end in a decimal point (i.e. `34.` must be changed to either `34.0` or `34`). Secondly, when using scientific notation, the `e` exponent must not immediately follow a decimal point (i.e. `3.e3` must be changed to either `3.0e3` or `3e3`). Finally, an empty string is no longer considered valid JSON.

Internal function failure on overflow

Previously, internal functions would silently truncate numbers produced from float-to-integer coercions when the float was too large to represent as an integer. Now, an E_WARNING will be emitted and ENTITY will be returned.

Fixes to custom session handler return values

Any predicate functions implemented by custom session handlers that return either ENTITY or `-1` will be fatal errors. If any value from these functions other than a boolean, `-1`, or `0` is returned, then it will fail and an E_WARNING will be emitted.

Sort order of equal elements

The internal sorting algorithm has been improved, what may result in different sort order of elements, which compare as equal, than before.

> Don't rely on the order of elements which compare as equal; it might change anytime.

Misplaced break and continue statements

`break` and `continue` statements outside of a loop or `switch` control structure are now detected at compile-time instead of run-time as before, and trigger an `E_COMPILE_ERROR`.

Constant disallowed as break and continue argument

`break` and `continue` statements no longer allow their argument to be a constant, and trigger a `E_COMPILE_ERROR`.

Mhash is not an extension anymore

The Mhash extension has been fully integrated into the Hash extension. Therefore, it is no longer possible to detect Mhash support with `extension_loaded`; use `function_exists` instead. Furthermore, Mhash is no longer reported by `get_loaded_extensions` and related features.

declare(ticks)

The declare(ticks) directive does no longer leak into different compilation units.

□□□ #2
□ □□□ □□□ 2026-10-31 17:33:06 UTC
□ □□□ □□□ 2026-10-31 17:33:08 UTC