Backward incompatible changes

- Throw on passing too few function arguments
- Forbid dynamic calls to scope introspection functions
- Invalid class, interface, and trait names
- Numerical string conversions now respect scientific notation
- Fixes to mt_rand algorithm
- rand aliased to mt_rand and srand aliased to mt_srand
- Disallow the ASCII delete control character in identifiers
- error_log changes with syslog value
- Do not call destructors on incomplete objects
- call_user_func handling of reference arguments
- The empty index operator is not supported for strings anymore
- Assignment via string index access on an empty string
- Removed ini directives
- Array ordering when elements are automatically created during by reference assignments has changed
- Sort order of equal elements
- Error message for E_RECOVERABLE errors
- $options parameter of unserialize()
- DateTime constructor incorporates microseconds
- Fatal errors to Error exceptions conversions
- Lexically bound variables cannot reuse names
- long2ip() parameter type change
- JSON encoding and decoding
- Changes to mb_ereg and mb_eregi parameter semantics
- Drop support for the sslv2 stream
- Forbid "return;" for typed returns already at compile-time

Throw on passing too few function arguments

Previously, a warning would be emitted for invoking user-defined functions with too few arguments. Now, this warning has been promoted to an Error exception. This change only applies to user-defined functions, not internal functions. For example:

```
<?php
function test($param){}
test();
```

ENTITY

```
Fatal error: Uncaught ArgumentCountError: Too few arguments to function test(), 0 passed in %s on line %d and exactly 1 expected in %s:%d
```

Forbid dynamic calls to scope introspection functions

Dynamic calls for certain functions have been forbidden (in the form of `$func()` or `array_map('extract', ...)`, etc). These functions either inspect or modify another scope, and present with them ambiguous and unreliable behavior. The functions are as follows:

- `assert` - with a string as the first argument
- `compact`
- `extract`
- `func_get_args`
- `func_get_arg`
- `func_num_args`
- `get_defined_vars`
- `mb_parse_str` - with one arg
- `parse_str` - with one arg

```
<?php
(function () {
    $func = 'func_num_args';
    $func();
})();
```

ENTITY

```
Warning: Cannot call func_num_args() dynamically in %s on line %d
```

Invalid class, interface, and trait names

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

- `void`
- `iterable`

Numerical string conversions now respect scientific notation

Integer operations and conversions on numerical strings now respect scientific notation. This also includes the `(int)` cast operation, and the following functions: `intval` (where the base is 10), `settype`, `decbin`, `decoct`, and `dechex`.

Fixes to `mt_rand` algorithm

`mt_rand` will now default to using the fixed version of the Mersenne Twister algorithm. If deterministic output from `mt_rand` was relied upon, then `MT_RAND_PHP` can be used as optional second parameter to `mt_srand` to preserve the old (incorrect) implementation.

`rand` aliased to `mt_rand` and `srand` aliased to `mt_srand`

`rand` and `srand` have now been made aliases to `mt_rand` and `mt_srand`, respectively. This means that the output for the following functions have changed: `rand`, `shuffle`, `str_shuffle`, and `array_rand`.

Disallow the ASCII delete control character in identifiers

The ASCII delete control character (`0x7F`) can no longer be used in identifiers that are not quoted.

`error_log` changes with `syslog` value

If the `error_log` ini setting is set to `syslog`, the PHP error levels are mapped to the syslog error levels. This brings finer differentiation in the error logs in contrary to the previous approach where all the errors are logged with the notice level only.

Do not call destructors on incomplete objects

Destructors are now never called for objects that throw an exception during the execution of their constructor. In previous versions this behavior depended on whether the object was referenced outside the constructor (e.g. by an exception backtrace).

`call_user_func` handling of reference arguments

`call_user_func` will now always generate a warning upon calls to functions that expect references as arguments. Previously this depended on whether the call was fully qualified.

Additionally, `call_user_func` and `call_user_func_array` will no longer abort the function call in this case. The "expected reference" warning will be emitted, but the call will proceed as usual.

The empty index operator is not supported for strings anymore

Applying the empty index operator to a string (e.g. `$str[] = $x`) throws a fatal error instead of converting silently to array.

Assignment via string index access on an empty string

String modification by character on an empty string now works like for non-empty strings, i.e. writing to an out of range offset pads the string with spaces, where non-integer types are converted to integer, and only the first character of the assigned string is used. Formerly, empty strings where silently treated like an empty array.

```
<?php
$a = '';
$a[10] = 'foo';
var_dump($a);
?>
```

ENTITY

```
array(1) {
  [10]=>
  string(3) "foo"
}
```

ENTITY

```
string(11) "          f"
```

Removed ini directives

The following ini directives have been removed:

- `session.entropy_file`
- `session.entropy_length`
- `session.hash_function`
- `session.hash_bits_per_character`

Array ordering when elements are automatically created during by reference assignments has changed

The order of the elements in an array has changed when those elements have been automatically created by referencing them in a by reference assignment. For example:

```
<?php
$array = [];
$array["a"] =& $array["b"];
$array["b"] = 1;
```

```
var_dump($array);
?>
```

ENTITY

```
array(2) {
  ["a"]=>
  &int(1)
  ["b"]=>
  &int(1)
}
```

ENTITY

```
array(2) {
  ["b"]=>
  &int(1)
  ["a"]=>
  &int(1)
}
```

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.

Error message for E_RECOVERABLE errors

The error message for E_RECOVERABLE errors has been changed from "Catchable fatal error" to "Recoverable fatal error".

$options parameter of unserialize()

The `allowed_classes` element of the $options parameter of `unserialize` is now strictly typed, i.e. if anything other than an `array` or a `bool` is given, unserialize() returns ENTITY and issues an `E_WARNING`.

DateTime constructor incorporates microseconds

`DateTime` and `DateTimeImmutable` now properly incorporate microseconds when constructed from the current time, either explicitly or with a relative string (e.g. `"first day of next month"`). This means that naive comparisons of two newly created instances will now more likely return ENTITY instead of ENTITY:

```
<?php
new DateTime() == new DateTime();
?>
```

Fatal errors to `Error` exceptions conversions

In the Date extension, invalid serialization data for `DateTime` or `DatePeriod` classes, or timezone initialization failure from serialized data, will now throw an `Error` exception from the __wakeup or __set_state methods, instead of resulting in a fatal error.

In the DBA extension, data modification functions (such as `dba_insert`) will now throw an `Error` exception instead of triggering a catchable fatal error if the key does not contain exactly two elements.

In the DOM extension, invalid schema or RelaxNG validation contexts will now throw an `Error` exception instead of resulting in a fatal error. Similarly, attempting to register a node class that does not extend the appropriate base class, or attempting to read an invalid property or write to a readonly property, will also now throw an `Error` exception.

In the IMAP extension, email addresses longer than 16385 bytes will throw an `Error` exception instead of resulting in a fatal error.

In the Intl extension, failing to call the parent constructor in a class extending `Collator` before invoking the parent methods will now throw an `Error` instead of resulting in a recoverable fatal error. Also, cloning a `Transliterator` object will now throw an `Error` exception on failure to clone the internal transliterator instead of resulting in a fatal error.

In the LDAP extension, providing an unknown modification type to `ldap_batch_modify` will now throw an `Error` exception instead of resulting in a fatal error.

In the mbstring extension, the `mb_ereg` and `mb_eregi` functions will now throw a `ParseError` exception if an invalid PHP expression is provided and the 'e' option is used.

In the Mcrypt extension, the `mcrypt_encrypt` and `mcrypt_decrypt` will now throw an `Error` exception instead of resulting in a fatal error if mcrypt cannot be initialized.

In the mysqli extension, attempting to read an invalid property or write to a readonly property will now throw an `Error` exception instead of resulting in a fatal error.

In the Reflection extension, failing to retrieve a reflection object or retrieve an object property will now throw an `Error` exception instead of resulting in a fatal error.

In the Session extension, custom session handlers that do not return strings for session IDs will now throw an `Error` exception instead of resulting in a fatal error when a function is called that must generate a session ID.

In the SimpleXML extension, creating an unnamed or duplicate attribute will now throw an `Error` exception instead of resulting in a fatal error.

In the SPL extension, attempting to clone an `SplDirectory` object will now throw an `Error` exception instead of resulting in a fatal error. Similarly, calling ArrayIterator::append when iterating over an object will also now throw an `Error` exception.

In the standard extension, the `assert` function, when provided with a string argument as its first parameter, will now throw a `ParseError` exception instead of resulting in a catchable fatal error if the PHP code is invalid. Similarly, calling `forward_static_call` outside of a class scope will now throw an `Error` exception.

In the Tidy extension, creating a `tidyNode` manually will now throw an `Error` exception instead of resulting in a fatal error.

In the WDDX extension, a circular reference when serializing will now throw an `Error` exception instead of resulting in a fatal error.

In the XML-RPC extension, a circular reference when serializing will now throw an instance of `Error` exception instead of resulting in a fatal error.

In the Zip extension, the ZipArchive::addGlob method will now throw an `Error` exception instead of resulting in a fatal error if glob support is not available.

Lexically bound variables cannot reuse names

Variables bound to a closure via the `use` construct cannot use the same name as any ENTITY, `$this`, or any parameter. For example, all of these function definition will result in a fatal error:

```
<?php
$f = function () use ($_SERVER) {};
$f = function () use ($this) {};
$f = function ($param) use ($param) {};
```

long2ip() parameter type change

`long2ip` now expects an `int` instead of a `string`.

JSON encoding and decoding

The `serialize_precision` ini setting now controls the serialization precision when encoding `float`s.

Decoding an empty key now results in an empty property name, rather than `_empty_` as a property name.

```
<?php
var_dump(json_decode(json_encode(['' => 1])));
```

ENTITY

```
object(stdClass)#1 (1) {
  [""]=>
  int(1)
}
```

When supplying the `JSON_UNESCAPED_UNICODE` flag to `json_encode`, the sequences U+2028 and U+2029 are now escaped.

Changes to `mb_ereg` and `mb_eregi` parameter semantics

The third parameter to the `mb_ereg` and `mb_eregi` functions (`regs`) will now be set to an empty array if nothing was matched. Formerly, the parameter would not have been modified.

Drop support for the sslv2 stream

The sslv2 stream has now been dropped in OpenSSL.

Forbid "return;" for typed returns already at compile-time

Return statements without argument in functions which declare a return type now trigger `E_COMPILE_ERROR` (unless the return type is declared as `void`), even if the return statement would never be reached.

□□□ #2
□ □□□ □□□ 2026-10-31 17:33:38 UTC
□ □□□ □□□ 2026-10-31 17:33:40 UTC