A SUMMARY OF THE CHANGES FROM PHP 5.6 TO PHP 8.2

TABLE OF CONTENTS

OVERALL FROM 5.6 - 8.2

BACKWARDS-INCOMPATIBLE CHANGES

- Several fatal errors have been converted into Exceptions, a tool to check for possible fatal errors before they execute. These exceptions extend the Error class in PHP, which further extend the Throwable class. Therefore, custom error handles may no longer be triggered due to an exception being thrown instead, causing fatal errors for said uncaught exceptions. The PHP 7 errors page contains further information on the behaviors of errors.
- The function set_exception_handler is no longer guaranteed to receive Exception objects, causing a fatal error when an exception handler initialized with said function using a type declaration of Exception tries to catch an Error object. To make the set_exception_handler() function work on PHP7, simply replace the exception type declaration with Throwable.
- Internal constructors always throw exceptions on failure, instead of an unusable object or null.
- Variable handling has been changed. Due to this, handling of indirections have also changed. The following chart should serve as a guide on the differences between PHP5's interpretation and PHP7's interpretation. Any intentions for a certain expression to utilize a PHP5 interpretation must be changed to accommodate to these new changes:

Expressions:	PHP5's interpretation:	PHP7+'s interpretation:
$$foo['bar']['baz']	${$foo['bar']['baz']}	($$foo)['bar']['baz']
$foo->$bar['baz']	$foo->{$bar['baz']}	($foo->$bar)['baz']
$foo->$bar['baz']()	$foo->{$bar['baz']}()	($foo->$bar)['baz']()
Foo::$bar['baz']()	Foo::{$bar['baz']}()	(Foo::$bar)['baz']()

- However, if you want to use the old interpretation, just explicitly write the expression in the PHP 5 interpretation format.
- The changes to indirection also affect the global keyword, but the curly brace syntax can be used to emulate the previous behavior if needed:
  Example:

```
<?php
function f() {
   // Valid in PHP 5 only.
   global $$foo->bar;
```

```
    // Valid in PHP 5 and 7.
    global ${$foo->bar};
}
?>
```

- list() handling has also been changed.
- list() no longer assigns the variables in reverse order, and now assigns variables in the order they are defined.
- list() constructs can no longer be empty. The following are illegal in PHP7 and beyond:

```
<?php
list() = $a;
list(,,) = $a;
list($x, list(), $y) = $a;
?>
```

- list() can no longer be used to convert strings into arrays. Use `str_split()` instead.
- Parentheses in function arguments (what you pass in to functions) no longer affect behavior, and the presence of parentheses in function arguments now generates warnings in PHP7. (Could be deprecated later on.) Instead, make sure to define new values with whatever you would place in the parentheses in function arguments, and then pass those values in the function arguments.
- Integer handling has been changed.
- Octal literals that contain invalid numbers (that is, octals that contain 8 and/or 9 as digits), now cause a parse error.
- Bit-shifts (>> and <<) by negative numbers now result in errors.
- Bit-shifts in both directions beyond 64 now always result in 0.
- Division by 0 has been changed. Consult the following table to learn of the distinctions.

Expression	PHP 5.6 Output	PHP 7+ Output
3 / 0	E_WARNING + false	float(INF)
0 / 0	E_WARNING + false	float(NAN)
0 % 0	E_WARNING + false	Fatal Error

- String handling has been changed.

- Hexadecimal numbers are no longer considered as numeric meaning they cannot be used wherever a numeric string can be used. Consult the following table for more information:

Code to run:	PHP 5.6 Output	PHP7+ Output:

<?php
var_dump("0x123" == "291");
var_dump(is_numeric("0x123"));
var_dump("0xe" + "0x1");
var_dump(substr("foo", "0x1"));
?>
	bool(true)
	bool(true)
	int(15)
	string(2) "oo"
		bool(false)
		bool(false)
		int(0)
		
		Notice: A non well formed numeric value encountered in /tmp/test.php on line 5
		string(3) "foo"

- `filter_var()` Can be used to check if a string contains a hex number, and also to convert a string of that type to an int.
- \u can cause errors, and should be removed.
- Several functions have been removed. A list of removed functions is provided:
    - call_user_method()
    - call_user_method_array()
    - All ereg* functions
    - Mcrypt aliases
    - All ext/mysql functions
    - All ext/mssql functions
    - Intl aliases
    - set_magic_quotes_runtime() and magic_quotes_runtime()
    - set_socket_blocking()
    - dl()
    - PostScript Type 1 fonts
- New objects cannot be assigned by reference (=&)
- Null, true, and false cannot be used to name classes, interfaces, and traits.
- Additionally, resource, object, mixed, and numeric should not be used to name classes, interfaces, and traits.
- Yield is now a right associative operator. Parentheses can be used to preserve the original intention.
- Functions can no longer have multiple parameters with the same name
- Functions inspecting arguments now report the current parameter value.
- Switch statements cannot have multiple default blocks

- $HTTP_RAW_POST_DATA has been removed, and the php://input stream should be used instead.
- The JSON extension is replaced with JSOND, causing 3 major changes
    1. Numbers can no longer end in a decimal point.
    2. When using scientific notation, exponents must not immediately follow a decimal point.
    3. Empty strings are no longer considered valid JSON.
- Prior to 7.1, you were able to pass a function with too few arguments, only scoring a warning. Now, this results in an Error exception being thrown. To avert this crisis, make sure to either create overloaded versions of functions that accommodate for a lower list of parameters, or just add default values in the function call. In layman's terms, overloaded functions share the same name, but have different parameter lists and/or different body code. To accommodate for less parameter cases, simply copy - paste the old function, omit a selected parameter in the parameter list, and initialize it in the body as a default value.
- Dynamic calls (or calls that either inspect or modify another class), are forbidden for the following functions:
    - 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 argument)
    - parse_str() (with one argument)
- Doing said dynamic calls will result in a warning (but will possibly later on will be promoted to Error exceptions and fatal errors)
- An example of a dynamic call can be found below:

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

- The following names cannot be used to name iterables, strings, and traits:
    - Void
    - iterable
- Integer operations and conversions on numerical strings now respect scientific notation.

- This also includes the (int) typecast operation, and conversions between decimal and any other format. A list of functions affected can be found below:
    - intval() (where the base is 10)
    - settype()
    - decbin()
    - decoct()
    - dechex()
- rand() is now aliased to mt_rand()
- srand() is now aliased to mt_srand().
- The preceding two changes also affect the following functions:
    - rand()
    - shuffle()
    - str_shuffle()
    - array_rand()
- The ASCII delete control character (0x7F) can no longer be used in identifiers.
- Calling destructors on incomplete objects is now forbidden, and destructors are never called on objects that throw an exception during their initialization.
- call_user_func() and call_user_func_array() now always generate warnings upon calls to functions that expect references as arguments.
- The empty index operator is now not supported for strings anymore, and trying to apply it to a string now throws a fatal error.
- Example of applying an empty index operator to a string:
  `$str[] = $x`
- String modification by character on empty strings now works like for non-empty strings. A table of the change in string modification is provided below.

Code	Output in PHP 7.0	Output in PHP 7.1+

<?php
$a = '';
$a[10] = 'foo';
var_dump($a);
?>
	array(1) {
	  [10]=>
	  string(3) "foo"
	}
		string(11) " f"

- As you can see, writing to a location out of range pads the offset with spaces, and only the first character of the assigned character is used.
- The following ini directives have been removed:
    - session.entropy_file
    - session.entropy_length
    - session.hash_function

    - session.hash_bits_per_character
- The ordering of elements in an array has changed when said elements have been automatically created by referencing them in by a reference assignment. Consult the following table for further information about this change:

Code	Output in PHP 7.0	Output in PHP 7.1+

<?php
$array = [];
$array["a"] =& $array["b"];
$array["b"] = 1;
var_dump($array);
?>
	array(2) {
	["a"]=>
	&int(1)
	["b"]=>
	&int(1)
	}
		array(2) {
		["b"]=>
		&int(1)
		["a"]=>
		&int(1)
		}

- The error message for E_RECOVERABLE errors is now “Recoverable Fatal Error”
- The $options parameter of unserialize() is now strictly typed, only allowing arrays and bools to be given.
- DateTime and DateTimeImmutable now incorporate microseconds, resulting in comparisons between two instances that are one microsecond apart producing false.
- long2ip() now expects an int instead of a string.
- Return statements without arguments in functions that declare a return type are now forbidden.
- number_format() no longer can be able to return -0.
- Integer keys from arrays can now be able to be converted to objects, when casting arrays to objects, and vice versa. Consider what is now available in the following 2 code snippets:

Code Snippet 1:

```
<?php

// array to object
$arr = [0 => 1];
$obj = (object)$arr;
var_dump(
   $obj,
   $obj->{'0'}, // now accessible
   $obj->{0} // now accessible
);
```

Code Snippet 2:

```
<?php
```

```php
// object to array
$obj = new class {
   public function __construct()
   {
      $this->{0} = 1;
   }
};
$arr = (array)$obj;
var_dump(
   $arr,
   $arr[0], // now accessible
   $arr['0'] // now accessible
);
```

- Passing null to the get_class() function now results in a E_WARNING instead of the name of the enclosing class. In order to produce the same results that would be achieved in prior versions, simply omit the argument.
- Attempting to count() non-countable types now results in an E_WARNING error.
- This also affects the alias function sizeof()
- Arrays and objects of classes extending the Countable interface are considered countable types.
- gettype(), when used on a closed resource, now results in a string of “resource (closed)” being returned (as opposed to a string of “unknown type”)
- Using the is_object() function on the __PHP_Incomplete_Class class now returns true.
- Unqualified references to undefined constraints now result in an E_WARNING error.
- The officially supported minimum Windows version is now Windows 7.
- The object name is now hard-reserved, prohibiting it from being used as a class, trait, or interface name.
- Support for NetWare is now removed.
- The bcmod() function no longer truncates fractional numbers to integers.
- The following hash functions no longer accept non-cryptographic hashes:
    - hash_hmac()
    - hash_hmac_file()
    - hash_pbkdf2()
    - hash_init()
- The sql.safe_mode ini setting is now removed.
- The zone element of the array resulting from the date_parse() and date_parse_from_format() now represents seconds instead of minutes.

- The names of incoming cookies are now no longer url-decoded.
- Heredoc/Nowdoc changes have been made, thus causing doc strings that contain the end label inside their body to cause syntax errors.
- Continue statements that target switch control flow structures now generate warnings. To absolve this issue, replace every mention of continue in the switch statement with continue 2.
- Array accesses, where an array extends ArrayAccess, using string literals no longer result in an implicit cast from string literal to integer.
- Static properties are no longer separated by reference assignments, where static properties could be overridden by a typecast in a child cast. The loophole has now been fixed.
- References returned by Array and Property Accesses are now immediately unwrapped, providing the value instead of the memory address.
- Support for BeOS has been dropped.
- Exceptions thrown due to automatic conversions of warnings into exceptions now work the same way as manually-thrown exceptions.
- TypeError now reports wrong types as int and bool, instead of integer and boolean.
- rsh/ssh logins are disabled by default. Use imap.enable_insecure_rsh if you want to enable them.
- Reflection export to string now uses int and bool instead of integer and boolean.
- Trying to use values of types null, bool, int, float, or resource as an array will now generate notices.
- The get_declared_classes() function no longer returns anonymous classes that have not been instantiated yet.
- Fn is now a reserved keyword.
- A <?php tag at the end of a file will now be interpreted as an opening PHP tag.
- Password-hashing algorithm identifiers are now nullable strings instead of ints.
- htmlentities() now raises a notice if it is used with an encoding for which only basic entity substitution is supported..
- Fread() and fwrite() now return false if the operation failed.
- Attempting to serialize a CURLFile class or a Reflection object now generates an exception.
- Calling var_dump() or a similar function on a DateTime or DateTimeImmutable object now no longer leaves behind accessible properties on the object.
- Calling get_object_vars() on an ArrayObject instance now always returns the properties of the ArrayObject itself. Previously, it returned the values of the wrapped array/object.

- String-to-number comparisons have been changed. Non-strict comparisons between a number and a string now work by casting the number to a string and then comparing the two strings. Consult the following table to learn more about this new behavior:

Comparison	PHP 7.4 evaluation	PHP 8.0+ evaluation
0 == “0”	true	true
0 == “0.0”	true	true
0 == “foo”	true	false
0 == “”	true	false
42 == “ 42”	true	true
42 == “42foo”	true	false

- Match is now a reserved keyword.
- Mixed is now a reserved word, and can no longer be used to name a class, trait, or an interface.
- Assertion failures now throw by default. If the old behavior is wanted, set assert.exception=0 in the ini settings.
- Methods with the same name as the class are no longer interpreted as constructors. The __construct() method should be used instead.
- The ability to call non-static methods statically has been removed.
- The (real) and (unset) casts have been removed.
- Due to the removal of the track_errors ini directive, php_errormsg is no longer available. You can use the error_get_last() function instead.
- each() has been removed, replaced with foreach and ArrayIterator.
- The ability to unbind this from proper closures that contain uses of this have also been removed.
- Any array that uses a number n as its first numeric key will use n + 1 for its next implicit key, even if n is negative.
- The default error reporting label is now E_ALL.
- Display_startup_errors is now enabled by default.
- #[is no longer interpreted as the start of a comment, as this syntax is now used for attributes. Use // as the start of comments instead.
- The @ operator will no longer silence fatal errors, and error handlers that expect error_reporting to be 0 when @ is used should be adjusted to use a mask check instead. Consult the following code snippet for more information:

```php
<?php
// Replace
function my_error_handler($err_no, $err_msg, $filename, $linenum) {
if (error_reporting() == 0) {
return false;
}
// ...
}

// With
function my_error_handler($err_no, $err_msg, $filename, $linenum) {
if (!(error_reporting() & $err_no)) {
return false;
}
// ...
}
?>
```

- As for replacing the functionality of the error suppression operator, @, wrapping the function call in a try catch is the suggested replacement. Example:

```
// Change this

$image = @imagecreatefromjpeg($imagePath);

if (!$image) {
  // handle failure
}

// To something like this

try {
  $image = imagecreatefromjpeg($imagePath);
} catch (\Throwable $e) {
  $hadError = true;
}

if (($hadError ?? false) || !$image) {
  // handle failure
}
```

- Also, please make sure that error messages are not displayed in production environments, as it can result in information leaks. Ensure that display_errors=Off is used in conjunction with error logging.
- Inheritance errors due to incompatible method signatures will now always generate a fatal error.
- The precedence of the concatenation operator now takes the same precedence as bit shifts (that is, after addition and subtraction).
- Arguments with a default value that resolves to null at runtime no longer implicitly marks the argument type as null.
- The following cases now result in an Error instead of a Warning:
    - Attempting to write to a property of a non-object
    - Attempting to append an element to an array for which the PHP_INT_MAX key is already used.
    - Attempting to use an invalid type (array or object) as an array key or string offset.
    - Attempting to write to an array index of a scalar value.
    - Attempting to unpack a non-array/Traversable.
    - Attempting to access unqualified constants which are undefined.
- The following Notices have been converted into Warnings:
    - Attempting to read an undefined variable.
    - Attempting to read an undefined property.
    - Attempting to read an unidentified array key.
    - Attempting to read a property of a non-object.
    - Attempting to access an array index of a non-array
    - Attempting to convert an array to string.
    - Attempting to use a resource as an array key.
    - Attempting to use null, a boolean, or a float as a string offset.
    - Attempting to read an out-of-bounds string offset.
    - Attempting to assign an empty string to a string offset.
    - Attempting to assign multiple bytes to a string offset.
- Unexpected characters in source files now result in a ParseError exception.
- The generated name for anonymous classes has changed, now including the name of the first parent or interface.
- Non-absolute trait method references in trait alias adaptations are now required to be unambiguous.
- The signature of abstract methods defined in traits is now checked against the implementing class method.

- The arithmetic and bitwise operators +, -, *, /, **, %, <<, >>, &, |, ^, ~, ++, -- now consistently throw a TypeError when one of the operands is an array, resource, or non-overloaded object.
- The only exception to the change above is the array + array merge operation, which is still supported.
- Float to string casting now always behaves local-independently.
- Support for deprecated curly braces for offset access has been removed.
- Applying the final modifier on a private method now produces a method unless the method is a constructor.
- If an object constructor throws or exit()s, the object destructor will no longer be called.
- Namespaced names can no longer contain whitespace (what is outputted when you press the spacebar).
- Nested ternaries now require explicit parentheses.
- Declaring a function called assert() inside a whitespace is no longer allowed.
- Several resource s have been migrated to objects.
- The ability to import case-insensitive constants from type libraries has been removed.
- mktime() and gmmktime() now require at least one argument. time() can still be used to get the current timestamp.
- read_exif_data() has been removed, replaced by exif_read_data().
- The functions image2wbmp(), png2wbmp(), and jpeg2wbmp() have been removed.
- gmp_random() has been removed. Use gmp_random_range() or gmp_random_bits() instead.
- The unused default_host argument of imap_headerinfo() has been removed.
- The imap_header() alias function of imap_headerinfo() has been removed.
- The functions ldap_sort(), ldap_control_paged_result() and ldap_control_paged_result_response() have been removed.
- The OCI-Lob class has been renamed to OCILob
- odbc_connect() no longer reuses connections.
- openssl_seal() and openssl_open() now require method to be passed.
- The default error handling mode has been changed from “silent” to “exceptions”.
- Reflection export() methods have been removed. Instead, reflection objects can be cast to string.
- SplFileObject::fgetss() has been removed.
- assert() will no longer evaluate string arguments, instead treating them like any other argument.
- parse_str() can no longer be used without specifying a result array.
- The string.strip_tags filter has been removed.
- fgetss() has been removed.

- Access to the $GLOBALS array is now subject to a number of restrictions. Write access to the entire $GLOBALS array is no longer supported, however, you can still have write access to individual array elements.
- When a method using static variables is inherited (but not overridden), the inherited method will now share static variables with the parent method. Consult the following code snippet on the next page for further information:

```php
<?php
class A {
   public static function counter() {
      static $counter = 0;
      $counter++;
      return $counter;
   }
}
class B extends A {}
var_dump(A::counter()); // int(1)
var_dump(A::counter()); // int(2)
var_dump(B::counter()); // int(3), previously int(1)
var_dump(B::counter()); // int(4), previously int(2)
?>
```

- This means that static variables in methods now behave the same way as static properties.
- An optional parameter specified before required parameters is now always treated as required, even when using named arguments. Failing to do so will result in an error being thrown.
- Most non-final internal methods now require overriding methods to declare a compatible return type.
- Readonly is a keyword now, but you can still use it as a function name. However, that practice could possibly soon be deprecated, then removed, so the best practice is to treat it as a reserved keyword.
- Never is now a reserved keyword, thus it can no longer be used to name a class, interface, or trait. It is also prohibited from being used in namespaces.
- The mysqlnd.fetch_data_copy INI directive has been removed.
- ext/mcrypt has been removed.
- The (real) cast is removed. Use (float) as well. Same goes for the is_real() function and the is_float() function, respectively.
- DateTime's createFromImmutable function and DateTimeImmutable's createFromMutable function now tentatively return static, instead of a DateTime object

or a DateTimeImmutable object, respectively. The object operator will no longer work with what is returned from this function, and the scope resolution operator (::), should be used to access the properties.

- Number symbols in relative formats no longer accept a plus sign followed by a minus sign, i.e. +-2
- The ODBC and PDO_ODBC extensions now escape the username and password in the case where both a connection string and a username/password are passed, and the string must be appended to.
- glob() now returns an empty array if all paths are restricted by open_basedir(), instead of returning false. Use the following code snippet to preserve the original result:

```
if(count(glob()) == 0){
Return false;
}else{
Return glob();
}
```

- FileSystemIterator::__construct(): To maintain its previous behavior, the FileSystemIterator::SKIP_DOTS constant must be explicitly set using the flags parameter.
- The following functions are no longer locale-sensitive, now performing ASCII case conversion, as if the locale was C. The MBString extension offers localized versions of these functions:
    - strtolower()
    - strtoupper()
    - stristr()
    - stripos()
    - strripos()
    - lcfirst()
    - ucfirst()
    - ucwords()
    - str_ireplace()
- str_split() now returns an empty array for an empty string, instead of an array containing a single empty string.
- ksort() and krsort() now do numeric string comparison under SORT_REGULAR using standard PHP 8 rules.
- var_export() no longer omits the leading backslash for exported classes.
- The following methods now export their signature:
    - SplFileInfo::_bad_state_ex()
    - SplFileObject::getCsvControl()
    - SplFileObject::fflush()

    - SplFileObject::ftell()
    - SplFileObject::fgetc()
    - SplFileObject::fpassthru()
- SplFileObject::hasChildren() now tentatively returns false, instead of bool
- SplFileObject::getChildren() now tentatively returns null
- GlobIterator now returns an empty array if all paths are restricted by open_basedir.

DEPRECATED FEATURES

- password_hash() salt option is now deprecated.
- Nested ternary operations must explicitly use parentheses to dictate the order of operations.
- Array and string offset access syntax using curly braces is now deprecated.
- Using parent inside of a class without a parent is deprecated.
- The money_format() function is deprecated.
- If a parameter with a default value is followed by a required parameter, the default value has no effect. This has been deprecated, and can be resolved by dropping the default value.
- Calling get_defined_functions() with exclude_disabled explicitly set to false is deprecated and now no longer has an effect.
- Sort comparisons that return true or false now throws a deprecation warning, and should be replaced with an implementation that returns an integer less than, equal to, or greater than 0. Consider the following code snippet as a guideline for changes to your code:

```php
<?php
// Replace
usort($array, fn($a, $b) => $a > $b);
// With
usort($array, fn($a, $b) => $a <=> $b);
?>
```

- Using an empty file as ZipArchive is deprecated.
- Serializable must now be implemented with __serialize() or __unserialize().
- Scalar types for built-in functions are nullable by default. This feature has been deprecated.
- The implicit conversion of float to int which leads to a loss in precision is now deprecated. This affects array keys, int type declarations in coercive mode, and operators working on ints.
- Calling a static method, or accessing a static property directly on a trait is deprecated.
- Returning a value which is not from an array from __sleep() now generates a diagnostic.

- data_sunrise() and date_sunset have been deprecated in favor of date_sun_info()
- strptime() has been deprecated. Use date_parse_from_format() instead (for locale-independent parsing), or IntlDateFormatter::parse() (for locale-dependent parsing).
- strftime() and gmstrftime() have been deprecated. Use date() instead (for locale-independent formatting), or IntlDateFormatter::format() (for locale-dependent formatting).
- The FILTER_SANITIZE_STRING and FILTER_SANITIZE_STRIPPED filters have been deprecated. The filter.default INI directive is also deprecated.
- The mhash(), mhash_keygen_s2k(), mhash_count(), mhash_get_block_size(), and mhash_get_hash_name() functions have been deprecated. Use the hash_*() functions instead.
- Calling IntlCalendar::roll() with a bool argument is deprecated. Use 1 and -1 instead of true and false, respectively.
- Calling key(), current(), next(), prev(), reset(), or end() on objects is deprecated. Either use get_mangled_object_vars() on the object first, or use ArrayIterator .
- The auto_detect_line_endings INI directive is deprecated. If you need to detect line endings, handle " \r" line breaks manually instead.
- The FILE_BINARY and FILE_TEXT constants have been deprecated. They never had any effect.
- The creation of dynamic properties is deprecated, unless the class opts in by using the #[\AllowDynamicProperties] attribute.
- Callables that are not accepted by the $callable() syntax are deprecated.
- The ${var} style of string interpolation is deprecated. Use "$var"/"{$var}" instead.
- Usage of the QPrint, Base64, Uuencode, and HTML-ENTITIES text encodings is deprecated for all MBString functions.
- utf8_encode() and utf8_decode() have been deprecated.

MISCELLANEOUS CHANGES

- We have replaced the mod_php module with FastCGI. Because of this, the following implications need to be kept in mind, including:
    - The directory in which the conf file for FastCGI is stored in Apache/mod_fcgi+php-fpm (as opposed to mod_php's Apache/mod_php)
    - The removal of support for mod_php will also incur the removal of the support of the usage of 'php_value' and 'php_flag' in .htaccess.
    - PHP logs for the FastCGI module are set in php-fpm configs

    - In order to add hotlink content (like js, css, images), Options +FollowSymLinks - Indexes must be set in .htaccess, while using the symlink for /var/www/html -> /UserData/AppData/webroot.
    - It can also be set as a directive for the whole web root in an apache conf file.

- The behavior of PHP PDO, when querying for a database, has changed for number-type columns. Consider the following code snippet for a more elaborate explanation:

In PHP 8.1, when querying a database, PHP PDO will now return integers for number type columns whereas in previous versions it returned strings. Either ensure your code won't break due to, for instance, triple equals checks, or set PDO to "stringify" all data coming from a database to keep the same behavior as PDO prior to PHP 8.1.

Example 1: Accept new PDO behavior and make sure triple equals still work

```
// Change this
$searchId = '1';
$isMatch = $idReturnedFromDb === $searchId;

// To this
$searchId = '1';
$isMatch = $idReturnedFromDb === (int) $searchId;
```

Example 2: Set PDO to cast all data to strings as it did prior to PHP 8.1

```
$dbConnection = new \PDO($dsn, $user, $pass);
$dbConnection->setAttribute(\PDO::ATTR_STRINGIFY_FETCHES, true);
```

BACK TO TOP

INDEPENDENT-VERSION CHANGES

5.6 - 7.0

BACKWARDS-INCOMPATIBLE CHANGES

- Several fatal errors have been converted into Exceptions, a tool to check for possible fatal errors before they execute. These exceptions extend the Error class in PHP, which further extend the Throwable class. Therefore, custom error handles may no longer be triggered due to an exception being thrown instead, causing fatal errors for said uncaught exceptions. The PHP 7 errors page contains further information on the behaviors of errors.
- The function set_exception_handler is no longer guaranteed to receive Exception objects, causing a fatal error when an exception handler initialized with said function using a type declaration of Exception tries to catch an Error object. To make the set_exception_handler() function work on PHP7, simply replace the exception type declaration with Throwable.
- Internal constructors always throw exceptions on failure, instead of an unusable object or null.
- Variable handling has been changed. Due to this, handling of indirections have also changed. The following chart should serve as a guide on the differences between PHP5's interpretation and PHP7's interpretation. Any intentions for a certain expression to utilize a PHP5 interpretation must be changed to accommodate to these new changes:

Expressions:	PHP 5.6's interpretation:	PHP 7.0's interpretation:
$$foo['bar']['baz']	${$foo['bar']['baz']}	($$foo)['bar']['baz']
$foo->$bar['baz']	$foo->{$bar['baz']}	($foo->$bar)['baz']
$foo->$bar['baz']()	$foo->{$bar['baz']}()	($foo->$bar)['baz']()
Foo::$bar['baz']()	Foo::{$bar['baz']}()	(Foo::$bar)['baz']()

- However, if you want to use the old interpretation, just explicitly write the expression in the PHP 5 interpretation format.
- The changes to indirection also affect the global keyword, but the curly brace syntax can be used to emulate the previous behavior if needed:
  Example:

```
<?php
function f() {
   // Valid in PHP 5 only.
```

```
    global $$foo->bar;

    // Valid in PHP 5 and 7.
    global ${$foo->bar};
}
?>
```

- list() handling has also been changed.
- list() no longer assigns the variables in reverse order, and now assigns variables in the order they are defined.
- list() constructs can no longer be empty. The following are illegal in PHP7 and beyond:

```
<?php
list() = $a;
list(,,) = $a;
list($x, list(), $y) = $a;
?>
```

- list() can no longer be used to convert strings into arrays. Use `str_split()` instead.
- Parentheses in function arguments (what you pass in to functions) no longer affect behavior, and the presence of parentheses in function arguments now generates warnings in PHP7. (Could be deprecated later on.) Instead, make sure to define new values with whatever you would place in the parentheses in function arguments, and then pass those values in the function arguments.
- Integer handling has been changed.
- Octal literals that contain invalid numbers (that is, octals that contain 8 and/or 9 as digits), now cause a parse error.
- Bit-shifts (>> and <<) by negative numbers now result in errors.
- Bit-shifts in both directions beyond 64 now always result in 0.
- Division by 0 has been changed. Consult the following table to learn of the distinctions.

Expression	PHP 5.6 Output	PHP 7.0 Output
3 / 0	E_WARNING + false	float(INF)
0 / 0	E_WARNING + false	float(NAN)
0 % 0	E_WARNING + false	Fatal Error

- String handling has been changed.

- Hexadecimal numbers are no longer considered as numeric meaning they cannot be used wherever a numeric string can be used. Consult the following table for more information:

Code to run:	PHP 5.6 Output	PHP 7.0 Output:

<?php
var_dump("0x123" == "291");
var_dump(is_numeric("0x123"));
var_dump("0xe" + "0x1");
var_dump(substr("foo", "0x1"));
?>
	bool(true)
	bool(true)
	int(15)
	string(2) "oo"
		bool(false)
		bool(false)
		int(0)
		
		Notice: A non well formed numeric value encountered in /tmp/test.php on line 5
		string(3) "foo"

- `filter_var()` Can be used to check if a string contains a hex number, and also to convert a string of that type to an int.
- \u can cause errors, and should be removed.
- Several functions have been removed. A list of removed functions is provided:
    - call_user_method()
    - call_user_method_array()
    - All ereg* functions
    - Mcrypt aliases
    - All ext/mysql functions
    - All ext/mssql functions
    - Intl aliases
    - set_magic_quotes_runtime() and magic_quotes_runtime()
    - set_socket_blocking()
    - dl()
    - PostScript Type 1 fonts
- New objects cannot be assigned by reference (=&)
- Null, true, and false cannot be used to name classes, interfaces, and traits.
- Additionally, resource, object, mixed, and numeric should not be used to name classes, interfaces, and traits.
- Yield is now a right associative operator. Parentheses can be used to preserve the original intention.
- Functions can no longer have multiple parameters with the same name
- Functions inspecting arguments now report the current parameter value.
- Switch statements cannot have multiple default blocks

- $HTTP_RAW_POST_DATA has been removed, and the php://input stream should be used instead.

- The JSON extension is replaced with JSOND, causing 3 major changes
    1. Numbers can no longer end in a decimal point.
    2. When using scientific notation, exponents must not immediately follow a decimal point.
    3. Empty strings are no longer considered valid JSON.

DEPRECATED FEATURES

- PHP4 style constructors(methods with the same name as the class they are defined in) are now deprecated.
- Static calls to non-static methods are deprecated.
- password_hash() salt option is now deprecated.
- The ldap_sort() function has been deprecated.
- The capture_session_meta SSL option has been deprecated.

7.0 - 7.1

BACKWARDS-INCOMPATIBLE CHANGES

- Prior to 7.1, you were able to pass a function with too few arguments, only scoring a warning. Now, this results in an Error exception being thrown. To avert this crisis, make sure to either create overloaded versions of functions that accommodate for a lower list of parameters, or just add default values in the function call. In layman's terms, overloaded functions share the same name, but have different parameter lists and/or different body code. To accommodate for less parameter cases, simply copy - paste the old function, omit a selected parameter in the parameter list, and initialize it in the body as a default value.
- Dynamic calls (or calls that either inspect or modify another class), are forbidden for the following functions:
    - 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 argument)
        - parse_str() (with one argument)
    - Doing said dynamic calls will result in a warning (but will possibly later on will be promoted to Error exceptions and fatal errors)
    - An example of a dynamic call can be found below:

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

    - The following names cannot be used to name iterables, strings, and traits:
        - Void
        - iterable
    - Integer operations and conversions on numerical strings now respect scientific notation.
    - This also includes the (int) typecast operation, and conversions between decimal and any other format. A list of functions affected can be found below:
        - intval() (where the base is 10)
        - settype()
        - decbin()
        - decoct()
        - dechex()
    - rand() is now aliased to mt_rand()
    - srand() is now aliased to mt_srand().
    - The preceding two changes also affect the following functions:
        - rand()
        - shuffle()
        - str_shuffle()
        - array_rand()
    - The ASCII delete control character (0x7F) can no longer be used in identifiers.
    - Calling destructors on incomplete objects is now forbidden, and destructors are never called on objects that throw an exception during their initialization.
    - call_user_func() and call_user_func_array() now always generate warnings upon calls to functions that expect references as arguments.
    - The empty index operator is now not supported for strings anymore, and trying to apply it to a string now throws a fatal error.

- Example of applying an empty index operator to a string:
  `$str[] = $x`
- String modification by character on empty strings now works like for non-empty strings. A table of the change in string modification is provided below.

Code	Output in PHP 7.0	Output in PHP 7.1

<?php
$a = '';
$a[10] = 'foo';
var_dump($a);
?>
	array(1) {
	[10]=>
	string(3) "foo"
	}
		string(11) " f"

- As you can see, writing to a location out of range pads the offset with spaces, and only the first character of the assigned character is used.
- The following ini directives have been removed:
  - session.entropy_file
  - session.entropy_length
  - session.hash_function
  - session.hash_bits_per_character

- The ordering of elements in an array has changed when said elements have been automatically created by referencing them in by a reference assignment. Consult the following table for further information about this change:

Code	Output in PHP 7.0	Output in PHP 7.1

<?php
$array = [];
$array["a"] =& $array["b"];
$array["b"] = 1;
var_dump($array);
?>
	array(2) {
	["a"]=>
	&int(1)
	["b"]=>
	&int(1)
	}
		array(2) {
		["b"]=>
		&int(1)
		["a"]=>
		&int(1)
		}

- The error message for E_RECOVERABLE errors is now “Recoverable Fatal Error”
- The $options parameter of unserialize() is now strictly typed, only allowing arrays and bools to be given.

- DateTime and DateTimeImmutable now incorporate microseconds, resulting in comparisons between two instances that are one microsecond apart producing false.
- long2ip() now expects an int instead of a string.
- Return statements without arguments in functions that declare a return type are now forbidden.

DEPRECATED FEATURES

- ext/mcrypt is deprecated, and will be removed in PHP 7.2
- The e pattern modifier has been deprecated for the following functions:
  - mb_ereg_replace()
  - mb_eregi_replace()

7.1 - 7.2

BACKWARDS-INCOMPATIBLE CHANGES

- number_format() no longer can be able to return -0.
- Integer keys from arrays can now be able to be converted to objects, when casting arrays to objects, and vice versa. Consider what is now available in the following 2 code snippets:

Code Snippet 1:

```
<?php

// array to object
$arr = [0 => 1];
$obj = (object)$arr;
var_dump(
   $obj,
   $obj->{'0'}, // now accessible
   $obj->{0} // now accessible
);
```

Code Snippet 2:

```
<?php

// object to array
$obj = new class {
```

```
    public function __construct()
    {
        $this->{0} = 1;
    }
};
$arr = (array)$obj;
var_dump(
    $arr,
    $arr[0], // now accessible
    $arr['0'] // now accessible
);
```

- Passing null to the get_class() function now results in a E_WARNING instead of the name of the enclosing class. In order to produce the same results that would be achieved in prior versions, simply omit the argument.
- Attempting to count() non-countable types now results in an E_WARNING error.
- This also affects the alias function sizeof()
- Arrays and objects of classes extending the Countable interface are considered countable types.
- gettype(), when used on a closed resource, now results in a string of “resource (closed)” being returned (as opposed to a string of “unknown type”)
- Using the is_object() function on the __PHP_Incomplete_Class class now returns true.
- Unqualified references to undefined constraints now result in an E_WARNING error.
- The officially supported minimum Windows version is now Windows 7.
- The object name is now hard-reserved, prohibiting it from being used as a class, trait, or interface name.
- Support for NetWare is now removed.
- The bcmod() function no longer truncates fractional numbers to integers.
- The following hash functions no longer accept non-cryptographic hashes:
    - hash_hmac()
    - hash_hmac_file()
    - hash_pbkdf2()
    - hash_init()
- The sql.safe_mode ini setting is now removed.
- The zone element of the array resulting from the date_parse() and date_parse_from_format() now represents seconds instead of minutes.
- The names of incoming cookies are now no longer url-decoded.

DEPRECATED FEATURES

- Unquoted strings (non-existent global constants took as strings of themselves) are now deprecated.
- The png2wbmp() and jpeg2wbmp() functions are now deprecated and will be removed in 8.0
- The __autoload function has been deprecated.
- The create_function() function has now been deprecated and should be replaced with uses of anonymous functions.
- The (unset) cast is now deprecated.
- Using parse_str() with out the second argument is now deprecated.
- each() has been deprecated.
- assert() with a string argument is now deprecated.
- The read_exif_data() alias of the exif_read_data() function is now deprecated, and the original function should be used instead.

7.2 - 7.3

BACKWARDS-INCOMPATIBLE CHANGES

- Heredoc/Nowdoc changes have been made, thus causing doc strings that contain the end label inside their body to cause syntax errors.
- Continue statements that target switch control flow structures now generate warnings. To absolve this issue, replace every mention of continue in the switch statement with continue 2.
- Array accesses, where an array extends ArrayAccess, using string literals no longer result in an implicit cast from string literal to integer.
- Static properties are no longer separated by reference assignments, where static properties could be overridden by a typecast in a child cast. The loophole has now been fixed.
- References returned by Array and Property Accesses are now immediately unwrapped, providing the value instead of the memory address.
- Support for BeOS has been dropped.
- Exceptions thrown due to automatic conversions of warnings into exceptions now work the same way as manually-thrown exceptions.
- TypeError now reports wrong types as int and bool, instead of integer and boolean.
- rsh/ssh logins are disabled by default. Use imap.enable_insecure_rsh if you want to enable them.
- Reflection export to string now uses int and bool instead of integer and boolean.

DEPRECATED FEATURES

- The declaration of case-sensitive constants is now deprecated.
- Declaring a function called assert() inside a namespace is now deprecated.
- The fgetss() function is now deprecated.
- image2wbmp() is now deprecated.

7.3 - 7.4

BACKWARDS-INCOMPATIBLE CHANGES

- Trying to use values of types null, bool, int, float, or resource as an array will now generate notices.
- The get_declared_classes() function no longer returns anonymous classes that have not been instantiated yet.
- Fn is now a reserved keyword.
- A <?php tag at the end of a file will now be interpreted as an opening PHP tag.
- Password-hashing algorithm identifiers are now nullable strings instead of ints.
- htmlentities() now raises a notice if it is used with an encoding for which only basic entity substitution is supported..
- Fread() and fwrite() now return false if the operation failed.
- Attempting to serialize a CURLFile class or a Reflection object now generates an exception.
- Calling var_dump() or a similar function on a DateTime or DateTimeImmutable object now no longer leaves behind accessible properties on the object.
- Calling get_object_vars() on an ArrayObject instance now always returns the properties of the ArrayObject itself. Previously, it returned the values of the wrapped array/object.

DEPRECATED FEATURES

- Nested ternary operations must explicitly use parentheses to dictate the order of operations.
- Array and string offset access syntax using curly braces is now deprecated.
- The (real) cast is deprecated. Use (float) as well. Same goes for the is_real() function and the is_float() function, respectively.
- Unbinding $this of a non-static closure that uses $this is deprecated.

- Using parent inside of a class without a parent is deprecated.
- The money_format() function is deprecated.

7.4 - 8.0

BACKWARDS-INCOMPATIBLE CHANGES

- String-to-number comparisons have been changed. Non-strict comparisons between a number and a string now work by casting the number to a string and then comparing the two strings. Consult the following table to learn more about this new behavior:

Comparison	PHP 7.4 evaluation	PHP 8.0 evaluation
0 == “0”	True	true
0 == “0.0”	True	true
0 == “foo”	True	false
0 == “”	True	false
42 == “ 42”	True	true
42 == “42foo”	True	false

- Match is now a reserved keyword.
- Mixed is now a reserved word, and can no longer be used to name a class, trait, or an interface.
- Assertion failures now throw by default. If the old behavior is wanted, set assert.exception=0 in the ini settings.
- Methods with the same name as the class are no longer interpreted as constructors. The `__construct()` method should be used instead.
- The ability to call non-static methods statically has been removed.
- The (real) and (unset) casts have been removed.
- Due to the removal of the track_errors ini directive, php_errormsg is no longer available. You can use the `error_get_last()` function instead.
- each() has been removed, replaced with `foreach` and ArrayIterator.
- The ability to unbind this from proper closures that contain uses of this have also been removed.
- Any array that uses a number n as its first numeric key will use n + 1 for its next implicit key, even if n is negative.

- The default error reporting label is now E_ALL.
- Display_startup_errors is now enabled by default.
- The @ operator will no longer silence fatal errors, and error handlers that expect error_reporting to be 0 when @ is used should be adjusted to use a mask check instead. Consult the following code snippet for more information:

```
<?php
// Replace
function my_error_handler($err_no, $err_msg, $filename, $linenum) {
if (error_reporting() == 0) {
return false;
}
// ...
}

// With
function my_error_handler($err_no, $err_msg, $filename, $linenum) {
if (!(error_reporting() & $err_no)) {
return false;
}
// ...
}
?>
```

- As for replacing the functionality of the error suppression operator, @, wrapping the function call in a try catch is the suggested replacement. Example:

```
 // Change this

$image = @imagecreatefromjpeg($imagePath);

if (!$image) {
  // handle failure
}

// To something like this

try {
  $image = imagecreatefromjpeg($imagePath);
} catch (\Throwable $e) {
  $hadError = true;
```

```
}

if (($hadError ?? false) || !$image) {
  // handle failure
}
```

- Also, please make sure that error messages are not displayed in production environments, as it can result in information leaks. Ensure that display_errors=Off is used in conjunction with error logging.
- #[is no longer interpreted as the start of a comment, as this syntax is now used for attributes. Use // as the start of comments instead.
- Inheritance errors due to incompatible method signatures will now always generate a fatal error.
- The precedence of the concatenation operator now takes the same precedence as bit shifts (that is, after addition and subtraction).
- Arguments with a default value that resolves to null at runtime no longer implicitly marks the argument type as null.
- The following cases now result in an Error instead of a Warning:
    - Attempting to write to a property of a non-object
    - Attempting to append an element to an array for which the PHP_INT_MAX key is already used.
    - Attempting to use an invalid type (array or object) as an array key or string offset.
    - Attempting to write to an array index of a scalar value.
    - Attempting to unpack a non-array/Traversable.
    - Attempting to access unqualified constants which are undefined.
- The following Notices have been converted into Warnings:
    - Attempting to read an undefined variable.
    - Attempting to read an undefined property.
    - Attempting to read an unidentified array key.
    - Attempting to read a property of a non-object.
    - Attempting to access an array index of a non-array
    - Attempting to convert an array to string.
    - Attempting to use a resource as an array key.
    - Attempting to use null, a boolean, or a float as a string offset.
    - Attempting to read an out-of-bounds string offset.
    - Attempting to assign an empty string to a string offset.
    - Attempting to assign multiple bytes to a string offset.
- Unexpected characters in source files now result in a ParseError exception.

- The generated name for anonymous classes has changed, now including the name of the first parent or interface.
- Non-absolute trait method references in trait alias adaptations are now required to be unambiguous.
- The signature of abstract methods defined in traits is now checked against the implementing class method.
- The arithmetic and bitwise operators `+, -, *, /, **, %, <<, >>, &, |, ^, ~, ++, --` now consistently throw a TypeError when one of the operands is an array, resource, or non-overloaded object.
- The only exception to the change above is the array + array merge operation, which is still supported.
- Float to string casting now always behaves local-independently.
- Support for deprecated curly braces for offset access has been removed.
- Applying the final modifier on a private method now produces a method unless the method is a constructor.
- If an object constructor throws or exit()s, the object destructor will no longer be called.
- Namespaced names can no longer contain whitespace (what is outputted when you press the spacebar).
- Nested ternaries now require explicit parentheses.
- Declaring a function called assert() inside a whitespace is no longer allowed.
- Several `resource` s have been migrated to objects.
- The ability to import case-insensitive constants from type libraries has been removed.
- mktime() and gmmktime() now require at least one argument. time() can still be used to get the current timestamp.
- read_exif_data() has been removed, replaced by `exif_read_data()`.
- The functions image2wbmp(), png2wbmp(), and jpeg2wbmp() have been removed.
- gmp_random() has been removed. Use `gmp_random_range()` or `gmp_random_bits()` instead.
- The unused default_host argument of imap_headerinfo() has been removed.
- The imap_header() alias function of imap_headerinfo() has been removed.
- The functions ldap_sort(), ldap_control_paged_result() and ldap_control_paged_result_response() have been removed.
- The OCI-Lob class has been renamed to OCILob
- odbc_connect() no longer reuses connections.
- openssl_seal() and openssl_open() now require method to be passed.
- The default error handling mode has been changed from “silent” to “exceptions”.
- Reflection export() methods have been removed. Instead, reflection objects can be cast to string.

- SplFileObject::fgetss() has been removed.
- assert() will no longer evaluate string arguments, instead treating them like any other argument.
- parse_str() can no longer be used without specifying a result array.
- The string.strip_tags filter has been removed.
- fgetss() has been removed.

DEPRECATED FEATURES

- If a parameter with a default value is followed by a required parameter, the default value has no effect. This has been deprecated, and can be resolved by dropping the default value.
- Calling get_defined_functions() with exclude_disabled explicitly set to false is deprecated and now no longer has an effect.
- Sort comparisons that return true or false now throws a deprecation warning, and should be replaced with an implementation that returns an integer less than, equal to, or greater than 0. Consider the following code snippet as a guideline for changes to your code:

```
<?php
// Replace
usort($array, fn($a, $b) => $a > $b);
// With
usort($array, fn($a, $b) => $a <=> $b);
?>
```

- Using an empty file as ZipArchive is deprecated.

8.0 - 8.1

BACKWARDS-INCOMPATIBLE CHANGES

- Access to the $GLOBALS array is now subject to a number of restrictions. Write access to the entire $GLOBALS array is no longer supported, however, you can still have write access to individual array elements.
- When a method using static variables is inherited (but not overridden), the inherited method will now share static variables with the parent method. Consult the following code snippet for further information:

```
<?php
```

```php
class A {
   public static function counter() {
      static $counter = 0;
      $counter++;
      return $counter;
   }
}
class B extends A {}
var_dump(A::counter()); // int(1)
var_dump(A::counter()); // int(2)
var_dump(B::counter()); // int(3), previously int(1)
var_dump(B::counter()); // int(4), previously int(2)
?>
```

- This means that static variables in methods now behave the same way as static properties.
- An optional parameter specified before required parameters is now always treated as required, even when using named arguments. Failing to do so will result in an error being thrown.
- Most non-final internal methods now require overriding methods to declare a compatible return type.
- Readonly is a keyword now, but you can still use it as a function name. However, that practice could possibly soon be deprecated, then removed, so the best practice is to treat it as a reserved keyword.
- Never is now a reserved keyword, thus it can no longer be used to name a class, interface, or trait. It is also prohibited from being used in namespaces.
- The mysqlnd.fetch_data_copy INI directive has been removed.

DEPRECATED FEATURES

- Serializable must now be implemented with __serialize() or __unserialize().
- Scalar types for built-in functions are nullable by default. This feature has been deprecated.
- The implicit conversion of float to int which leads to a loss in precision is now deprecated. This affects array keys, int type declarations in coercive mode, and operators working on ints.
- Calling a static method, or accessing a static property directly on a trait is deprecated.
- Returning a value which is not from an array from __sleep() now generates a diagnostic.
- data_sunrise() and date_sunset have been deprecated in favor of date_sun_info()

- strptime() has been deprecated. Use date_parse_from_format() instead (for locale-independent parsing), or IntlDateFormatter::parse() (for locale-dependent parsing).
- strftime() and gmstrftime() have been deprecated. Use date() instead (for locale-independent formatting), or IntlDateFormatter::format() (for locale-dependent formatting).
- The FILTER_SANITIZE_STRING and FILTER_SANITIZE_STRIPPED filters have been deprecated. The filter.default INI directive is also deprecated.
- The mhash(), mhash_keygen_s2k(), mhash_count(), mhash_get_block_size(), and mhash_get_hash_name() functions have been deprecated. Use the hash_*() functions instead.
- Calling IntlCalendar::roll() with a bool argument is deprecated. Use 1 and -1 instead of true and false, respectively.
- Calling key(), current(), next(), prev(), reset(), or end() on objects is deprecated. Either use get_mangled_object_vars() on the object first, or use ArrayIterator .
- The auto_detect_line_endings INI directive is deprecated. If you need to detect line endings, handle " \r" line breaks manually instead.
- The FILE_BINARY and FILE_TEXT constants have been deprecated. They never had any effect.

8.1 - 8.2

BACKWARDS-INCOMPATIBLE CHANGES

- DateTime's createFromImmutable function and DateTimeImmutable's createFromMutable function now tentatively return static, instead of a DateTime object or a DateTimeImmutable object, respectively. The object operator will no longer work with what is returned from this function, and the scope resolution operator (::), should be used to access the properties.
- Number symbols in relative formats no longer accept a plus sign followed by a minus sign, i.e. +-2
- The ODBC and PDO_ODBC extensions now escape the username and password in the case where both a connection string and a username/password are passed, and the string must be appended to.
- glob() now returns an empty array if all paths are restricted by open_basedir(), instead of returning false. Use the following code snippet to preserve the original result:
  if(count(glob()) == 0){
  Return false;
  }else{

Return glob();
}

- FileSystemIterator::__construct(): To maintain its previous behavior, the FileSystemIterator::SKIP_DOTS constant must be explicitly set using the flags parameter.
- The following functions are no longer locale-sensitive, now performing ASCII case conversion, as if the locale was C. The MBString extension offers localized versions of these functions:
    - strtolower()
    - strtoupper()
    - stristr()
    - stripos()
    - strripos()
    - lcfirst()
    - ucfirst()
    - ucwords()
    - str_ireplace()
- str_split() now returns an empty array for an empty string, instead of an array containing a single empty string.
- ksort() and krsort() now do numeric string comparison under SORT_REGULAR using standard PHP 8 rules.
- var_export() no longer omits the leading backslash for exported classes.
- The following methods now export their signature:
    - SplFileInfo::_bad_state_ex()
    - SplFileObject::getCsvControl()
    - SplFileObject::fflush()
    - SplFileObject::ftell()
    - SplFileObject::fgetc()
    - SplFileObject::fpassthru()
- SplFileObject::hasChildren() now tentatively returns false, instead of bool
- SplFileObject::getChildren() now tentatively returns null
- GlobIterator now returns an empty array if all paths are restricted by open_basedir.

DEPRECATED FEATURES

- The creation of dynamic properties is deprecated, unless the class opts in by using the #[\AllowDynamicProperties] attribute.
- Callables that are not accepted by the $callable() syntax are deprecated.
- The ${var} style of string interpolation is deprecated. Use “$var”/”{$var}” instead.

- Usage of the QPrint, Base64, Uuencode, and HTML-ENTITIES text encodings is deprecated for all MBString functions.
- utf8_encode() and utf8_decode() have been deprecated.

ADDITIONAL CHANGES

- We have replaced the mod_php module with FastCGI. Because of this, the following implications need to be kept in mind, including:
    - The directory in which the conf file for FastCGI is stored in Apache/mod_fcgi+php-fpm (as opposed to mod_php's Apache/mod_php)
    - The removal of support for mod_php will also incur the removal of the support of the usage of 'php_value' and 'php_flag' in .htaccess.
    - PHP logs for the FastCGI module are set in php-fpm configs
    - In order to add hotlink content (like js, css, images), Options +FollowSymLinks -Indexes must be set in .htaccess, while using the symlink for /var/www/html -> /UserData/AppData/webroot.
    - It can also be set as a directive for the whole web root in an apache conf file.

- External dependencies have been updated
- The error handler signature has been changed.
- cURL library added SSL host verification
- Curl_init returns CurlHandle instead of resource
- Warnings occur when counting non-countable types.
- Create_function has been removed, and an anonymous function must be used instead.
- Optional parameters before required parameters are not allowed anymore.

BACK TO TOP

Source: PHP Migration notes 5->8

- Migrating from PHP 8.0.x to PHP 8.1.x
- Migrating from PHP 7.4.x to PHP 8.0.x
- Migrating from PHP 7.3.x to PHP 7.4.x
- Migrating from PHP 7.2.x to PHP 7.3.x
- Migrating from PHP 7.1.x to PHP 7.2.x
- Migrating from PHP 7.0.x to PHP 7.1.x
- Migrating from PHP 5.6.x to PHP 7.0.x