Runtime Configuration

The behaviour of these functions is affected by settings in `php.ini`.

PHP Options/Info Configuration Options	
Name	assert.active
Default	"1"
Changeable	INI_ALL
Changelog	Deprecated as of PHP 8.3.0

Name	assert.bail
Default	"0"
Changeable	INI_ALL
Changelog	Deprecated as of PHP 8.3.0

Name	assert.warning
Default	"1"
Changeable	INI_ALL
Changelog	Deprecated as of PHP 8.3.0

Name	assert.callback
Default	NULL
Changeable	INI_ALL
Changelog	Deprecated as of PHP 8.3.0

Name	assert.quiet_eval
Default	"0"
Changeable	INI_ALL
Changelog	Removed as of PHP 8.0.0

Name	assert.exception
Default	"1"
Changeable	INI_ALL
Changelog	Prior to PHP 8.0.0, defaults to "0". Deprecated as of PHP 8.3.0

Name	enable_dl
Default	"1"
Changeable	INI_SYSTEM
Changelog	This deprecated feature will certainly be removed in the future.

Name	max_execution_time
Default	"30"
Changeable	INI_ALL
Changelog	

Name	max_input_time
Default	"-1"
Changeable	INI_PERDIR
Changelog	

Name	max_input_nesting_level
Default	"64"
Changeable	INI_PERDIR
Changelog	

Name	max_input_vars
Default	"1000"
Changeable	INI_PERDIR
Changelog	

Name	zend.enable_gc
Default	"1"
Changeable	INI_ALL
Changelog	

Name	zend.max_allowed_stack_size
Default	"0"
Changeable	INI_SYSTEM
Changelog	Available as of PHP 8.3.0.

Name	zend.reserved_stack_size
Default	"0"
Changeable	INI_SYSTEM
Changelog	Available as of PHP 8.1.0.

Name	fiber.stack_size
Default	
Changeable	INI_ALL
Changelog	

Available as of PHP 8.1.0.

For further details and definitions of the PHP_* modes, see the Where a configuration setting may be set.
Here's a short explanation of the configuration directives.

`assert.active` bool
Enable assert() evaluation. zend.assertions should be used instead to control the behaviour of assert().

Warning This feature has been *DEPRECATED* as of PHP 8.3.0. Relying on this feature is highly discouraged.

`assert.bail` bool
Terminate script execution on failed assertions.

Warning This feature has been *DEPRECATED* as of PHP 8.3.0. Relying on this feature is highly discouraged.

`assert.warning` bool
Issue a *PHP* warning for each failed assertion.

Warning This feature has been *DEPRECATED* as of PHP 8.3.0. Relying on this feature is highly discouraged.

`assert.callback` string
User function to call on failed assertions.

Warning This feature has been *DEPRECATED* as of PHP 8.3.0. Relying on this feature is highly discouraged.

`assert.quiet_eval` bool

Warning This feature was *REMOVED* as of PHP 8.0.0.

Use the current setting of error_reporting() during assertion expression evaluation. If enabled, no errors are shown (implicit error_reporting(0)) while evaluation. If disabled, errors are shown according to the settings of error_reporting().

`assert.exception` bool
Issue an AssertionError exception for the failed assertion.

Warning This feature has been *DEPRECATED* as of PHP 8.3.0. Relying on this feature is highly discouraged.

`enable_dl` bool
This directive allows to turn dynamic loading of PHP extensions with dl() on and off.

The main reason for turning dynamic loading off is security. With dynamic loading, it's possible to ignore all open_basedir restrictions. The default is to allow dynamic loading.

`max_execution_time` int
This sets the maximum time in seconds a script is allowed to run before it is terminated by the parser. This helps prevent poorly written scripts from tying up the server. The default setting is 30. When running *PHP* from the command line the default setting is 0.

On non-Windows systems, the maximum execution time is not affected by system calls, stream operations etc. Please see the set_time_limit() function for more details.

Your web server can have other timeout configurations that may also interrupt *PHP* execution. Apache has a `Timeout` directive and IIS has a CGI timeout function. Both default to 300 seconds. See your web server documentation for specific details.

`max_input_time` int
This sets the maximum time in seconds a script is allowed to parse input data, like POST and GET. Timing begins at the moment PHP is invoked at the server and ends when execution begins. The default setting is `-1`, which means that max_execution_time is used instead. Set to 0 to allow unlimited time.

`max_input_nesting_level` int
Sets the max nesting depth of input variables (i.e. `$_GET`, `$_POST`).

`max_input_vars` int
How many input variables may be accepted (limit is applied to `$_GET`, `$_POST` and `$_COOKIE` superglobal separately). Use of this directive mitigates the possibility of denial of service attacks which use hash collisions. If there are more input variables than specified by this directive, an `E_WARNING` is issued, and further input variables are truncated from the request.

`zend.enable_gc` bool
Enables or disables the circular reference collector.

`zend.max_allowed_stack_size` int
The maximum native stack space that the operating system allows the program to consume. Trying to consume more than the operating system allows typically results in a hard crash with no easily available debugging information. To make debugging easier, the engine throws an Error before it happens (when the program uses more than zend.max_allowed_stack_size - zend.reserved_stack_size bytes of stack).

Recursion in user-defined code does not consume native stack space. However, internal functions and magic methods do. Very deep recursion involving these functions can cause the program to exhaust all available native stack space.

Possible values for this parameter are:

- 0: Auto-detect the maximum native stack space that the operating system allows the program to consume. This is the default. When detection is not possible, a known system default is used.
- -1: Disables stack size checking in the engine.
- Positive integer: A fixed size, in bytes. Setting this value too high has the same effect as disabling stack size checking.

As the stack size of fibers is determined by fiber.stack_size, the value of this parameter is used instead of zend.max_allowed_stack_size when checking stack usage during the execution of a Fiber.

Note:
This is not related to stack *buffer* overflows, and is not a security feature.

`zend.reserved_stack_size` int
The reserved stack size, in bytes. This is subtracted from the max allowed stack size, as a buffer, when checking the stack size.

Possible values for this parameter are:

- 0: Auto-detect a sensible size.
- Positive integer: A fixed size, in bytes.

`fiber.stack_size` int
The native stack size, in bytes, allocated for each Fiber.

The default value is 1MiB on systems with a pointer size lower than 8 bytes, or 2MiB otherwise.

Found A Problem?

Learn How To Improve This Page • Submit a Pull Request • Report a Bug

User Contributed Notes

+ add a note

There are no user contributed notes for this page.