php

- php://stdin, php://stdout and php://stderr
- php://input
- php://output
- php://fd
- php://memory and php://temp
- php://filter

php://

Accessing various I/O streams

ENTITY

PHP provides a number of miscellaneous I/O streams that allow access to PHP's own input and output streams, the standard input, output and error file descriptors, in-memory and disk-backed temporary file streams, and filters that can manipulate other file resources as they are read from and written to.

php://stdin, php://stdout and php://stderr

`php://stdin`, `php://stdout` and `php://stderr` allow direct access to the corresponding input or output stream of the PHP process. The stream references a duplicate file descriptor, so if you open `php://stdin` and later close it, you close only your copy of the descriptor-the actual stream referenced by `STDIN` is unaffected. It is recommended that you simply use the constants `STDIN`, `STDOUT` and `STDERR` instead of manually opening streams using these wrappers.

`php://stdin` is read-only, whereas `php://stdout` and `php://stderr` are write-only.

php://input

`php://input` is a read-only stream that allows you to read raw data from the request body. It can be opened and read as many times as needed. `php://input` is not available in POST requests with `enctype="multipart/form-data"` if enable_post_data_reading option is enabled.

php://output

`php://output` is a write-only stream that allows you to write to the output buffer mechanism in the same way as `print` and `echo`.

php://fd

`php://fd` allows direct access to the given file descriptor. For example, `php://fd/3` refers to file descriptor 3.

php://memory and php://temp

`php://memory` and `php://temp` are read-write streams that allow temporary data to be stored in a file-like wrapper. One difference between the two is that `php://memory` will always store its data in memory, whereas `php://temp` will use a temporary file once the amount of data stored hits a predefined limit (the default is 2 MB). The location of this temporary file is determined in the same way as the `sys_get_temp_dir` function.

The memory limit of `php://temp` can be controlled by appending `/maxmemory:NN`, where `NN` is the maximum amount of data to keep in memory before using a temporary file, in bytes.

> Some PHP extensions may require a standard IO stream, and may attempt to cast a given stream to a standard IO stream. This cast can fail for memory streams as it requires the C `fopencookie` function to be available. This C function is *not* available on Windows.

php://filter

`php://filter` is a kind of meta-wrapper designed to permit the application of filters to a stream at the time of opening. This is useful with all-in-one file functions such as `readfile`, `file`, and `file_get_contents` where there is otherwise no opportunity to apply a filter to the stream prior the contents being read.

The `php://filter` target takes the following parameters as part of its path. Multiple filter chains can be specified on one path. Please refer to the examples for specifics on using these parameters.

Name	Description
`resource=ENTITYstream to be filteredENTITY`	This parameter is required. It specifies the stream that you would like to filter.
`read=ENTITYfilter list to apply to read chainENTITY`	This parameter is optional. One or more filter names can be provided here, separated by the pipe character (`
`write=ENTITYfilter list to apply to write chainENTITY`	This parameter is optional. One or more filter names can be provided here, separated by the pipe character (`
`ENTITYfilter list to apply to both chainsENTITY`	Any filter lists which are not prefixed by `read=` or `write=` will be applied to both the read and write chains as appropriate.

php://filter parameters {role="stream_wrapper"}

ENTITY

Attribute	Supported
Restricted by allow_url_fopen	No
Restricted by allow_url_include	`php://input`, `php://stdin`, `php://memory` and `php://temp` only.
Allows Reading	`php://stdin`, `php://input`, `php://fd`, `php://memory` and `php://temp` only.
Allows Writing	`php://stdout`, `php://stderr`, `php://output`, `php://fd`, `php://memory` and `php://temp` only.
Allows Appending	`php://stdout`, `php://stderr`, `php://output`, `php://fd`, `php://memory` and `php://temp` only. (Equivalent to writing)
Allows Simultaneous Reading and Writing	`php://fd`, `php://memory` and `php://temp` only.
Supports `stat`	No. However, `php://memory` and `php://temp` support `fstat`.
Supports `unlink`	No
Supports `rename`	No
Supports `mkdir`	No

Attribute	Supported
Supports `rmdir`	No
Supports `stream_select`	`php://stdin`, `php://stdout`, `php://stderr`, `php://fd` and `php://temp` only.

Wrapper Summary (for `php://filter`, refer to the summary of the wrapper being filtered) {role="stream_wrapper"}

ENTITY

php://temp/maxmemory

This optional parameter allows setting the memory limit before `php://temp` starts using a temporary file.

```
<?php
// Set the limit to 5 MB.
$fiveMBs = 5 * 1024 * 1024;
$fp = fopen("php://temp/maxmemory:$fiveMBs", 'r+');

fputs($fp, "hello\n");

// Read what we have written.
rewind($fp);
echo stream_get_contents($fp);
?>
```

php://filter/resource=ENTITYstream to be filteredENTITY

This parameter must be located at the end of your `php://filter` specification and should point to the stream which you want filtered.

```
<?php
/* This is equivalent to simply:
  readfile("http://www.example.com");
  since no filters are actually specified */

readfile("php://filter/resource=http://www.example.com");
```

```
?>
```

php://filter/read=ENTITYfilter list to apply to read chainENTITY

This parameter takes one or more filternames separated by the pipe character `|`.

```
<?php
/* This will output the contents of
  www.example.com entirely in uppercase */
readfile("php://filter/read=string.toupper/resource=http://www.example.com");

/* This will do the same as above
  but will also ROT13 encode it */
readfile("php://filter/read=string.toupper|string.rot13/resource=http://www.example.com");
?>
```

php://filter/write=ENTITYfilter list to apply to write chainENTITY

This parameter takes one or more filternames separated by the pipe character `|`.

```
<?php
/* This will filter the string "Hello World"
  through the rot13 filter, then write to
  example.txt in the current directory */
file_put_contents("php://filter/write=string.rot13/resource=example.txt","Hello World");
?>
```

php://memory and php://temp are not reusable

`php://memory` and `php://temp` are not reusable, i.e. after the streams have been closed there is no way to refer to them again.

```
<?php
file_put_contents('php://memory', 'PHP');
echo file_get_contents('php://memory'); // prints nothing
```

php://input to read JSON data from the request body

This example demonstrates how to read raw JSON data from POST, PUT and PATCH requests using `php://input`.

```
<?php
$input = file_get_contents("php://input");
$json_array = json_decode(
  json: $input,
  associative: true,
  flags: JSON_THROW_ON_ERROR
);

echo "Received JSON data: ";
print_r($json_array);
?>
```

□□□ #2
□ □□□ □□□ 2026-10-31 17:48:43 UTC
□ □□□ □□□ 2026-10-31 17:48:45 UTC