

php://

php:// — Accessing various I/O streams

Descrição ¶

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. Note that PHP exhibited buggy behavior in this regard until PHP 5.2.1. 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. In the case of POST requests, it is preferable to use *php://input* instead of [\\$HTTP_RAW_POST_DATA](#) as it does not depend on special *php.ini* directives. Moreover, for those cases where [\\$HTTP_RAW_POST_DATA](#) is not populated by default, it is a potentially less memory intensive alternative to activating [always populate raw post data](#). *php://input* is not available with *enctype="multipart/form-data"*.

Nota: A stream opened with *php://input* can only be read once; the stream does not support seek operations. However, depending on the SAPI implementation, it may be possible to open another *php://input* stream and restart reading. This is only possible if the request body data has been saved. Typically, this is the case for POST requests, but not other request methods, such as PUT or PROPFIND.

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. The only 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.

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.

php://filter parameters

Name	Description
<i>resource=<stream to be filtered></i>	This parameter is required. It specifies the stream that you would like to filter.
<i>read=<filter list to apply to read chain></i>	This parameter is optional. One or more filter names can be provided here, separated by the pipe character ().
<i>write=<filter list to apply to write chain></i>	This parameter is optional. One or more filter names can be provided here, separated by the pipe character ().
<i><filter list to apply to both chains></i>	Any filter lists which are not prefixed by <i>read=</i> or <i>write=</i> will be applied to both the read and write chains as appropriate.

Opções ¶

Wrapper Summary (for *php://filter*, refer to the summary of the wrapper being filtered)

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

Changelog ¶

Versão Descrição

5.3.6 *php://fd* was added.

5.1.0 *php://memory* and *php://temp* were added.

5.0.0 *php://filter* was added.

Exemplos ¶

Exemplo #1 *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);
?>
```

Exemplo #2 *php://filter/resource=<stream to be filtered>*

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");
?>
```

Exemplo #3 *php://filter/read=<filter list to apply to read chain>*

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");
?>
```

Exemplo #4 *php://filter/write=<filter list to apply to write chain>*

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");`
?>
 [add a note](#)

User Contributed Notes

There are no user contributed notes for this page.