# Zlib Compression

# Introduction

This module enables you to transparently read and write gzip (.gz) compressed files, through versions of most of the filesystem functions which work with gzip-compressed files (and uncompressed files, too, but not with sockets).

**Note**

Version 4.0.4 introduced a fopen-wrapper for .gz-files, so that you can use a special *zlib:* URL to access compressed files transparently using the normal f*() file access functions if you prefix the filename or path with *zlib:* when calling fopen(). This feature requires a C runtime library that provides the *fopencookie()* function. Up to now the GNU libc seems to be the only library that provides this feature.

In PHP 4.3.0, *zlib:* has been changed to *compress.zlib://* to prevent ambiguities with filenames containing ':' characters. The *fopencookie()* function is not longer required. More information is available in the section about Compression Streams.

# Installing/Configuring

## Requirements

This module uses the functions of » zlib by Jean-loup Gailly and Mark Adler. You have to use a zlib version >= 1.0.9 with this module.

## Installation

Zlib support in PHP is not enabled by default. You will need to configure PHP *--with-zlib[=DIR]*

The Windows version of PHP has built-in support for this extension. You do not need to load any additional extensions in order to use these functions.

| **Note** |
| --- |
| Built-in support for zlib on Windows is available with PHP 4.3.0. |

## Runtime Configuration

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

The zlib extension offers the option to transparently compress your pages on-the-fly, if the requesting browser supports this. Therefore there are three options in the configuration file *php.ini*.

### Zlib Configuration Options

| Name | Default | Changeable | Changelog |
| --- | --- | --- | --- |
| zlib.output_compression | "0" | PHP_INI_ALL | Available since PHP 4.0.5. |
| zlib.output_compression_level | "-1" | PHP_INI_ALL | Available since PHP 4.3.0. |
| zlib.output_handler | "" | PHP_INI_ALL | Available since PHP 4.3.0. |

For further details and definitions of the PHP_INI_* constants, see the php.ini directives.

Here's a short explanation of the configuration directives.

*zlib.output_compression* boolean / integer

Whether to transparently compress pages. If this option is set to "On" in *php.ini* or the Apache configuration, pages are compressed if the browser sends an "Accept-Encoding: gzip" or "deflate" header. "Content-Encoding: gzip" (respectively "deflate") and "Vary: Accept-Encoding" headers are added to the output. In runtime, it can be set only before sending any output. This option also accepts integer values instead of boolean "On"/"Off", using this you can set the output buffer size (default is 4KB).

| **Note** |
| --- |
| output_handler must be empty if this is set 'On' ! Instead you must use zlib.output_handler. |

*zlib.output_compression_level* integer

Compression level used for transparent output compression.

*zlib.output_handler* string

You cannot specify additional output handlers if zlib.output_compression is activated here. This setting does the same as output_handler but in a different order.

## Resource Types

This extension defines a file pointer resource returned by gzopen().

# Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

**FORCE_GZIP** ( integer )

**FORCE_DEFLATE** ( integer )

# Examples

This example opens a temporary file and writes a test string to it, then it prints out the content of this file twice.

**Example #1 - Small Zlib Example**

```
<?php

$filename = tempnam('/tmp', 'zlibtest') . '.gz';
echo "<html>\n<head></head>\n<body>\n<pre>\n";
$s = "Only a test, test, test, test, test, test, test, test!\n";

// open file for writing with maximum compression
$zp = gzopen($filename, "w9");

// write string to file
gzwrite($zp, $s);

// close file
gzclose($zp);

// open file for reading
$zp = gzopen($filename, "r");

// read 3 char
echo gzread($zp, 3);

// output until end of the file and close it.
gzpassthru($zp);
gzclose($zp);

echo "\n";

// open file and print content (the 2nd time).
if (readgzfile($filename) != strlen($s)) {
       echo "Error with zlib functions!";
}
unlink($filename);
echo "</pre>\n</body>\n</html>\n";

?>
```

# Zlib Functions

# gzclose

gzclose -- Close an open gz-file pointer

## Description

bool **gzclose** ( resource `$zp` )

Closes the given gz-file pointer.

## Parameters

*zp*
: The gz-file pointer. It must be valid, and must point to a file successfully opened by gzopen().

## Return Values

Returns **TRUE** on success or **FALSE** on failure.

## Examples

**Example #2 - gzclose() example**

```
<?php
$gz = gzopen('somefile.gz','w9');
gzputs ($gz, 'I was added to somefile.gz');
gzclose($gz);
?>
```

## See Also

- gzopen()

# gzcompress

gzcompress -- Compress a string

## Description

string **gzcompress** ( string `$data` [, int `$level` ] )

This function compress the given string using the *ZLIB* data format.

For details on the ZLIB compression algorithm see the document " » ZLIB Compressed Data Format Specification version 3.3 " (RFC 1950).

| **Note** |
| --- |
| This is *not* the same as gzip compression, which includes some header data. See gzencode() for gzip compression. |

## Parameters

*`data`*
: The data to compress.

*`level`*
: The level of compression. Can be given as 0 for no compression up to 9 for maximum compression.

## Return Values

The compressed string or **FALSE** if an error occurred.

## Examples

**Example #3 - gzcompress() example**

```
<?php
$compressed = gzcompress('Compress me', 9);
echo $compressed;
?>
```

## See Also

- gzdeflate()
- gzinflate()
- gzuncompress()
- gzencode()

# gzdecode

gzdecode -- Decodes a gzip compressed string

## Description

string **gzdecode** ( string `$data` [, int `$length` ] )

This function returns a decoded version of the input *`data`*.

## Parameters

*`data`*
: The data to decode, encoded by gzencode().

*`length`*
: The maximum length of data to decode.

## Return Values

The decoded string, or **FALSE** if an error occurred.

## See Also

- gzencode()

# gzdeflate

gzdeflate -- Deflate a string

## Description

string **gzdeflate** ( string `$data` [, int `$level` ] )

This function compress the given string using the *DEFLATE* data format.

For details on the DEFLATE compression algorithm see the document " » DEFLATE Compressed Data Format Specification version 1.3 " (RFC 1951).

## Parameters

*`data`*
: The data to deflate.

*`level`*
: The level of compression. Can be given as 0 for no compression up to 9 for maximum compression. If not given, the default compression level will be the default compression level of the zlib library.

## Return Values

The deflated string or **FALSE** if an error occurred.

## Examples

**Example #4 - gzdeflate() example**

```
<?php
$compressed = gzdeflate('Compress me', 9);
echo $compressed;
?>
```

## See Also

- gzinflate()
- gzcompress()
- gzuncompress()

- gzencode()

# gzencode

gzencode -- Create a gzip compressed string

## Description

string **gzencode** ( string `$data` [, int `$level` [, int `$encoding_mode` ] ] )

This function returns a compressed version of the input *`data`* compatible with the output of the *gzip* program.

For more information on the GZIP file format, see the document: » GZIP file format specification version 4.3 (RFC 1952).

## Parameters

*`data`*
: The data to encode.

*`level`*
: The level of compression. Can be given as 0 for no compression up to 9 for maximum compression. If not given, the default compression level will be the default compression level of the zlib library.

*`encoding_mode`*
: The encoding mode. Can be **FORCE_GZIP** (the default) or **FORCE_DEFLATE**. If you use **FORCE_DEFLATE**, you get a standard zlib deflated string (inclusive zlib headers) after the gzip file header but without the trailing crc32 checksum.

## Return Values

The encoded string, or **FALSE** if an error occurred.

## ChangeLog

| Version | Description |
|---|---|
| 4.2.0 | The *`encoding_mode`* parameter was added |

## Examples

The resulting data contains the appropriate headers and data structure to make a standard

.gz file, e.g.:

**Example #5 - Creating a gzip file**

```
<?php
$data = implode("", file("bigfile.txt"));
$gzdata = gzencode($data, 9);
$fp = fopen("bigfile.txt.gz", "w");
fwrite($fp, $gzdata);
fclose($fp);
?>
```

## See Also

- gzdecode()
- gzdeflate()
- gzinflate()
- gzuncompress()
- gzcompress()

# gzeof

gzeof -- Test for end-of-file on a gz-file pointer

## Description

int **gzeof** ( resource `$zp` )

Tests the given GZ file pointer for EOF (end-of-file).

## Parameters

*zp*
: The gz-file pointer. It must be valid, and must point to a file successfully opened by gzopen().

## Return Values

Returns **TRUE** if the gz-file pointer is at EOF or an error occurs; otherwise returns **FALSE**.

## Examples

**Example #6 - gzeof() example**

```
<?php
$gz = gzopen('somefile.gz', 'r');
while (!gzeof($gz)) {
 echo gzgetc($gz);
}
gzclose($gz);
?>
```

# gzfile

gzfile -- Read entire gz-file into an array

## Description

array **gzfile** ( string `$filename` [, int `$use_include_path` ] )

This function is identical to readgzfile(), except that it returns the file in an array.

## Parameters

*`filename`*
The file name.

*`use_include_path`*
You can set this optional parameter to *1*, if you want to search for the file in the include_path too.

## Return Values

An array containing the file, one line per cell.

## Examples

**Example #7 - gzfile() example**

```
<?php
$lines = gzfile('somefile.gz');
foreach ($lines as $line) {
   echo $line;
}
?>
```

## See Also

- readgzfile()
- gzopen()

# gzgetc

gzgetc -- Get character from gz-file pointer

## Description

string **gzgetc** ( resource `$zp` )

Returns a string containing a single (uncompressed) character read from the given gz-file pointer.

## Parameters

*zp*
: The gz-file pointer. It must be valid, and must point to a file successfully opened by gzopen().

## Return Values

The uncompressed character or **FALSE** on EOF (unlike gzeof() ).

## Examples

**Example #8 - gzgetc() example**

```
<?php
$gz = gzopen('somefile.gz', 'r');
while (!gzeof($gz)) {
 echo gzgetc($gz);
}
gzclose($gz);
?>
```

## See Also

- gzopen()
- gzgets()

# gzgets

gzgets -- Get line from file pointer

## Description

string **gzgets** ( resource `$zp`, int `$length` )

Gets a (uncompressed) string of up to length - 1 bytes read from the given file pointer. Reading ends when length - 1 bytes have been read, on a newline, or on EOF (whichever comes first).

## Parameters

*`zp`*
The gz-file pointer. It must be valid, and must point to a file successfully opened by gzopen().

*`length`*
The length of data to get.

## Return Values

The uncompressed string, or **FALSE** on error.

## Examples

**Example #9 - gzgets() example**

```
<?php
$handle = gzopen('somefile.gz', 'r');
while (!gzeof($handle)) {
  $buffer = gzgets($handle, 4096);
  echo $buffer;
}
gzclose($handle);
?>
```

## See Also

- gzopen()

- gzgetc()
- gzwrite()

# gzgetss

gzgetss -- Get line from gz-file pointer and strip HTML tags

## Description

string **gzgetss** ( resource `$zp`, int `$length` [, string `$allowable_tags` ] )

Identical to gzgets(), except that gzgetss() attempts to strip any HTML and PHP tags from the text it reads.

## Parameters

*`zp`*
The gz-file pointer. It must be valid, and must point to a file successfully opened by gzopen().

*`length`*
The length of data to get.

*`allowable_tags`*
You can use this optional parameter to specify tags which should not be stripped.

## Return Values

The uncompressed and striped string, or **FALSE** on error.

## ChangeLog

| Version | Description |
| --- | --- |
| 3.0.13 and 4.0.0 | *`allowable_tags`* was added. |

## Examples

**Example #10 - gzgetss() example**

```
<?php
$handle = gzopen('somefile.gz', 'r');
while (!gzeof($handle)) {
  $buffer = gzgetss($handle, 4096);
```

```
  echo $buffer;
}
gzclose($handle);
?>
```

## See Also

- gzopen()
- gzgets()
- strip_tags()

# gzinflate

gzinflate -- Inflate a deflated string

## Description

string **gzinflate** ( string `$data` [, int `$length` ] )

This function inflate a deflated string.

## Parameters

*`data`*
The data compressed by gzdeflate().

*`length`*
The maximum length of data to decode.

## Return Values

The original uncompressed data or **FALSE** on error.

The function will return an error if the uncompressed data is more than 32768 times the length of the compressed input *`data`* or more than the optional parameter *`length`*.

## Examples

**Example #11 - gzinflate() example**

```
<?php
$compressed   = gzdeflate('Compress me', 9);
$uncompressed = gzinflate($compressed);
echo $uncompressed;
?>
```

## See Also

- gzdeflate()
- gzcompress()
- gzuncompress()
- gzencode()

# gzopen

gzopen -- Open gz-file

## Description

resource **gzopen** ( string `$filename`, string `$mode` [, int `$use_include_path` ] )

Opens a gzip (.gz) file for reading or writing.

gzopen() can be used to read a file which is not in gzip format; in this case gzread() will directly read from the file without decompression.

## Parameters

*`filename`*
: The file name.

*`mode`*
: As in fopen() ( *rb* or *wb* ) but can also include a compression level ( *wb9* ) or a strategy: *f* for filtered data as in *wb6f*, *h* for *Huffman only compression* as in *wb1h*. (See the description of deflateInit2 in *zlib.h* for more information about the strategy parameter.)

*`use_include_path`*
: You can set this optional parameter to *1*, if you want to search for the file in the include_path too.

## Return Values

Returns a file pointer to the file opened, after that, everything you read from this file descriptor will be transparently decompressed and what you write gets compressed.

If the open fails, the function returns **FALSE**.

## Examples

**Example #12 - gzopen() Example**

```
<?php
$fp = gzopen("/tmp/file.gz", "r");
?>
```

## See Also

- gzclose()

# gzpassthru

gzpassthru -- Output all remaining data on a gz-file pointer

## Description

int **gzpassthru** ( resource `$zp` )

Reads to EOF on the given gz-file pointer from the current position and writes the (uncompressed) results to standard output.

| **Note** |
| --- |
| You may need to call gzrewind() to reset the file pointer to the beginning of the file if you have already written data to it. |

| **Tip** |
| --- |
| If you just want to dump the contents of a file to the output buffer, without first modifying it or seeking to a particular offset, you may want to use the readgzfile() function, which saves you the gzopen() call. |

## Parameters

*zp*
: The gz-file pointer. It must be valid, and must point to a file successfully opened by gzopen().

## Return Values

The number of uncompressed characters read from *gz* and passed through to the input, or **FALSE** on error.

## Examples

**Example #13 - gzpassthru() example**

```
<?php
$fp = gzopen('file.gz', 'r');
gzpassthru($fp);
gzclose($fp);
```

```
?>
```

# gzputs

gzputs -- Alias of gzwrite()

## Description

This function is an alias of: gzwrite().

# gzread

gzread -- Binary-safe gz-file read

## Description

string **gzread** ( resource `$zp`, int `$length` )

gzread() reads up to *`length`* bytes from the given gz-file pointer. Reading stops when *`length`* (uncompressed) bytes have been read or EOF is reached, whichever comes first.

## Parameters

*`zp`*
: The gz-file pointer. It must be valid, and must point to a file successfully opened by gzopen().

*`length`*
: The number of bytes to read.

## Return Values

The data that have been read.

## Examples

**Example #14 - gzread() example**

```
<?php
// get contents of a gz-file into a string
$filename = "/usr/local/something.txt.gz";
$zd = gzopen($filename, "r");
$contents = gzread($zd, 10000);
gzclose($zd);
?>
```

## See Also

- gzwrite()
- gzopen()
- gzgets()

- gzgetss()
- gzfile()
- gzpassthru()

# gzrewind

gzrewind -- Rewind the position of a gz-file pointer

## Description

bool **gzrewind** ( resource `$zp` )

Sets the file position indicator of the given gz-file pointer to the beginning of the file stream.

## Parameters

*zp*
: The gz-file pointer. It must be valid, and must point to a file successfully opened by gzopen().

## Return Values

Returns **TRUE** on success or **FALSE** on failure.

## See Also

- gzseek()
- gztell()

# gzseek

gzseek -- Seek on a gz-file pointer

## Description

int **gzseek** ( resource `$zp`, int `$offset` )

Sets the file position indicator for the given file pointer to the given offset byte into the file stream. Equivalent to calling (in C) *gzseek(zp, offset, SEEK_SET)*.

If the file is opened for reading, this function is emulated but can be extremely slow. If the file is opened for writing, only forward seeks are supported; gzseek() then compresses a sequence of zeroes up to the new starting position.

## Parameters

*`zp`*
The gz-file pointer. It must be valid, and must point to a file successfully opened by gzopen().

*`offset`*
The seeked offset.

## Return Values

Upon success, returns 0; otherwise, returns -1. Note that seeking past EOF is not considered an error.

## Examples

**Example #15 - gzseek() example**

```
<?php
$gz = gzopen('somefile.gz', 'r');
gzseek($gz,2);
echo gzgetc($gz);
gzclose($gz);
?>
```

## See Also

- gztell()
- gzrewind()

# gztell

gztell -- Tell gz-file pointer read/write position

## Description

int **gztell** ( resource `$zp` )

Gets the position of the given file pointer; i.e., its offset into the uncompressed file stream.

## Parameters

*zp*
: The gz-file pointer. It must be valid, and must point to a file successfully opened by gzopen().

## Return Values

The position of the file pointer or **FALSE** if an error occurs.

## See Also

- gzopen()
- gzseek()
- gzrewind()

# gzuncompress

gzuncompress -- Uncompress a compressed string

## Description

string **gzuncompress** ( string `$data` [, int `$length` ] )

This function uncompress a compressed string.

## Parameters

*`data`*
The data compressed by gzcompress().

*`length`*
The maximum length of data to decode.

## Return Values

The original uncompressed data or **FALSE** on error.

The function will return an error if the uncompressed data is more than 32768 times the length of the compressed input *`data`* or more than the optional parameter *`length`*.

## Examples

**Example #16 - gzuncompress() example**

```
<?php
$compressed   = gzcompress('Compress me', 9);
$uncompressed = gzuncompress($compressed);
echo $uncompressed;
?>
```

## See Also

- gzcompress()
- gzinflate()
- gzdeflate()
- gzencode()

# gzwrite

gzwrite -- Binary-safe gz-file write

## Description

int **gzwrite** ( resource `$zp`, string `$string` [, int `$length` ] )

gzwrite() writes the contents of *`string`* to the given gz-file.

## Parameters

*`zp`*
The gz-file pointer. It must be valid, and must point to a file successfully opened by gzopen().

*`string`*
The string to write.

*`length`*
The number of uncompressed bytes to write. If supplied, writing will stop after *`length`* (uncompressed) bytes have been written or the end of *`string`* is reached, whichever comes first.

| **Note** |
| --- |
| Note that if the *`length`* argument is given, then the magic_quotes_runtime configuration option will be ignored and no slashes will be stripped from *`string`*. |

## Return Values

Returns the number of (uncompressed) bytes written to the given gz-file stream.

## Examples

**Example #17 - gzwrite() example**

```
<?php
$string = 'Some information to compress';
$gz = gzopen('somefile.gz','w9');
gzwrite($gz, $string);
gzclose($gz);
?>
```

## See Also

- gzread()
- gzopen()

# readgzfile

readgzfile -- Output a gz-file

## Description

int **readgzfile** ( string `$filename` [, int `$use_include_path` ] )

Reads a file, decompresses it and writes it to standard output.

readgzfile() can be used to read a file which is not in gzip format; in this case readgzfile() will directly read from the file without decompression.

## Parameters

*`filename`*
: The file name. This file will be opened from the filesystem and its contents written to standard output.

*`use_include_path`*
: You can set this optional parameter to *1*, if you want to search for the file in the include_path too.

## Return Values

Returns the number of (uncompressed) bytes read from the file. If an error occurs, **FALSE** is returned and unless the function was called as *@readgzfile*, an error message is printed.

## See Also

- gzpassthru()
- gzfile()
- gzopen()

# zlib_get_coding_type

zlib_get_coding_type -- Returns the coding type used for output compression

## Description

string **zlib_get_coding_type** ( void )

Returns the coding type used for output compression.

## Return Values

Possible return values are *gzip*, *deflate*, or **FALSE**.

## See Also

- The zlib.output_compression directive