## NAME

phar, phar.phar – PHAR (PHP archive) command line tool

## SYNOPSIS

**phar** <command> [options] ...

## DESCRIPTION

The **PHAR** file format provides a way to put entire PHP applications into a single file called a "phar" (PHP Archive) for easy distribution and installation.

With the **phar** command you can create, update or extract PHP archives.

Commands: add compress delete extract help help-list info list meta-del meta-get meta-set pack sign stub-get stub-set tree version

### add command

Add entries to a PHAR package.

Required arguments:

**-f** *file* Specifies the phar *file* to work on.

**...** Any number of input files and directories. If -i is in use then ONLY files and matching the given regular expression are being packed. If -x is given then files matching that regular expression are NOT being packed.

Optional arguments:

**–a** *alias* Provide an *alias* name for the phar file.

**–c** *algo* Compression algorithm (see **COMPRESSION** )

**–i** *regex* Specifies a regular expression for input files.

**–l** *level* Number of preceding subdirectories to strip from file entries

**–x** *regex* Regular expression for input files to exclude.

### compress command

Compress or uncompress all files or a selected entry.

Required arguments:

**–c** *algo* Compression algorithm (see **COMPRESSION** )

**-f** *file* Specifies the phar *file* to work on.

Optional arguments:

**-e** *entry* Name of *entry* to work on (must include PHAR internal directory name if any).

### delete command

Delete entry from a PHAR archive

Required arguments:

**–e** *entry* Name of *entry* to work on (must include PHAR internal directory name if any).

**-f** *file* Specifies the phar *file* to work on.

### extract command

Extract a PHAR package to a directory.

Required arguments:

**-f** *file* Specifies the phar *file* to work on.

Optional arguments:

**-i** *regex* Specifies a regular expression for input files.

**-x** *regex* Regular expression for input files to exclude.

**...** Directory to extract to (defaults to '.').

### help command

This help or help for a selected command.

Optional arguments:

**...** Optional command to retrieve help for.

### help-list command

Lists available commands.

### info command

Get information about a PHAR package.

By using -k it is possible to return a single value.

Required arguments:

**-f** *file* Specifies the phar *file* to work on.

Optional arguments:

**-k** *index* Subscription *index* to work on.

### list command

List contents of a PHAR archive.

Required arguments:

**-f** *file* Specifies the phar *file* to work on.

Optional arguments:

**-i** *regex* Specifies a regular expression for input files.

**-x** *regex* Regular expression for input files to exclude.

### meta-del command

Delete meta information of a PHAR entry or a PHAR package.

If -k is given then the metadata is expected to be an array and the given index is being deleted.

If something was deleted the return value is 0 otherwise it is 1.

Required arguments:

**-f** *file* Specifies the phar *file* to work on.

Optional arguments:

**-e** *entry* Name of *entry* to work on (must include PHAR internal directory name if any).

**-k** *index* Subscription *index* to work on.

### meta-get command

Get meta information of a PHAR entry or a PHAR package in serialized from. If no output file is specified for meta data then stdout is being used. You can also specify a particular index using -k. In that case the metadata is expected to be an array and the value of the given index is returned using echo rather than using serialize. If that index does not exist or no meta data is present then the return value is 1.

Required arguments:

**-f** *file* Specifies the phar *file* to work on.

Optional arguments:

**-e** *entry* Name of *entry* to work on (must include PHAR internal directory name if any).

**-k** *index* Subscription *index* to work on.

### meta-set command

Set meta data of a PHAR entry or a PHAR package using serialized input. If no input file is specified for meta data then stdin is being used. You can also specify a particular index using -k. In that case the metadata is expected to be an array and the value of the given index is being set. If the metadata is not present or empty a new array will be created. If the metadata is present and a flat value then the return value is 1. Also using -k the input is been taken directly rather then being serialized.

Required arguments:

**-f** *file* Specifies the phar *file* to work on.

**-m** *meta* Meta data to store with entry (serialized php data).

Optional arguments:

**-e** *entry* Name of *entry* to work on (must include PHAR internal directory name if any).

**-k** *index* Subscription *index* to work on.

### pack command

Pack files into a PHAR archive.

When using -s <stub>, then the stub file is being excluded from the list of input files/dirs.To create an archive that contains PEAR class PHP_Archive then point -p argument to PHP/Archive.php.

Required arguments:

**-f** *file* Specifies the phar *file* to work on.

**...** Any number of input files and directories. If -i is in use then ONLY files and matching the given regular expression are being packed. If -x is given then files matching that regular expression are NOT being packed.

Optional arguments:

**−a** *alias* Provide an *alias* name for the phar file.

**−b** *bang* Hash-bang line to start the archive (e.g. #!/usr/bin/php). The hash mark itself '#!' and the newline character are optional.

**−c** *algo* Compression algorithm (see **COMPRESSION** )

**−h** *hash* Selects the *hash* algorithm (see **HASH** )

**−i** *regex* Specifies a regular expression for input files.

**−l** *level* Number of preceding subdirectories to strip from file entries

**−p** *loader* Location of PHP_Archive class file (pear list-files PHP_Archive).You can use '0' or '1' to locate it automatically using the mentioned pear command. When using '0' the command does not error out when the class file cannot be located. This switch also adds

some code around the stub so that class PHP_Archive gets registered as phar:// stream wrapper if necessary. And finally this switch will add the file phar.inc from this package and load it to ensure class Phar is present.

**−s** *stub* Select the *stub* file.

**−x** *regex* Regular expression for input files to exclude.

**−y** *key* Private *key* for OpenSSL signing.

### sign command

Set signature hash algorithm.

Required arguments:

**-f** *file* Specifies the phar *file* to work on.

**−h** *hash* Selects the *hash* algorithm (see **HASH** )

Optional arguments:

**−y** *key* Private *key* for OpenSSL signing.

### stub-get command

Get the stub of a PHAR file. If no output file is specified as stub then stdout is being used.

Required arguments:

**-f** *file* Specifies the phar *file* to work on.

Optional arguments:

**−s** *stub* Select the *stub* file.

### stub-set command

Set the stub of a PHAR file. If no input file is specified as stub then stdin is being used.

Required arguments:

**-f** *file* Specifies the phar *file* to work on.

Optional arguments:

**−b** *bang* Hash-bang line to start the archive (e.g. #!/usr/bin/php). The hash mark itself '#!' and the newline character are optional.

**−p** *loader* Location of PHP_Archive class file (pear list-files PHP_Archive).You can use '0' or '1' to locate it automatically using the mentioned pear command. When using '0' the command does not error out when the class file cannot be located. This switch also adds some code around the stub so that class PHP_Archive gets registered as phar:// stream wrapper if necessary. And finally this switch will add the file phar.inc from this package and load it to ensure class Phar is present.

**−s** *stub* Select the *stub* file.

### tree command

Get a directory tree for a PHAR archive.

Required arguments:

**-f** *file* Specifies the phar *file* to work on.

Optional arguments:

**−i** *regex* Specifies a regular expression for input files.

**−x** *regex* Regular expression for input files to exclude.

**version command**
Get information about the PHAR environment and the tool version.

## COMPRESSION

Algorithms:

**0** No compression

**none** No compression

**auto** Automatically select compression algorithm

**gz** GZip compression

**gzip** GZip compression

**bz2** BZip2 compression

**bzip2** BZip2 compression

## HASH

Algorithms:

**md5** MD5

**sha1** SHA1

**sha256** SHA256

**sha512** SHA512

**openssl** OpenSSL

## SEE ALSO

For a more or less complete description of PHAR look here:
**http://php.net/phar**

## BUGS

You can view the list of known bugs or report any new bug you found at:
**http://bugs.php.net**

## AUTHORS

The PHP Group: Thies C. Arntzen, Stig Bakken, Andi Gutmans, Rasmus Lerdorf, Sam Ruby, Sascha Schumann, Zeev Suraski, Jim Winstead, Andrei Zmievski.

Work for the PHP archive was done by Gregory Beaver, Marcus Boerger.

A List of active developers can be found here:
**http://www.php.net/credits.php**

And last but not least PHP was developed with the help of a huge amount of contributors all around the world.

## VERSION INFORMATION

This manpage describes **phar**, version 7.3.11-0ubuntu0.19.10.1.

## COPYRIGHT