

NAME

`tlmgr` – the TeX Live Manager

SYNOPSIS

`tlmgr` [*option*]... *action* [*option*]... [*operand*]...

DESCRIPTION

tlmgr manages an existing TeX Live installation, both packages and configuration options. It performs many of the same actions as **texconfig**(1), and more besides. (**texconfig** continues to be included and supported, but **tlmgr** is now preferred.)

The most up-to-date version of this documentation is available at <http://tug.org/texlive/tlmgr.html>, along with procedures for updating **tlmgr** itself, disaster recovery when it's broken, and test versions.

OPTIONS

The following options to **tlmgr** are global options not specific to any action. Although we present **tlmgr** invocation in the above order, all options can be given at any place and in arbitrary order. The first non-option argument will be the main action. In all cases, `--option` and `-option` are equivalent, and an `=` is optional between an option name and its value.

--repository *url|path*

Specifies the package repository from which packages should be installed or updated, overriding the default package repository found in the installation's TeX Live Package Database (TLPDB, a file named `texlive.tlpdb`). The documentation for `install-tl` has more details about this (<http://tug.org/texlive/doc/install-tl.html>).

For backward compatibility and convenience, `--location` and `--repo` are accepted as aliases for this option.

--gui [*action*]

You can give this option together with an action to be brought directly into the respective screen of the GUI. For example, running

```
tlmgr --gui update
```

starts you directly at the update screen.

--gui-lang *llcode*

Normally the GUI tries to deduce your language from the environment (on Windows via the registry, on Unix via `LC_MESSAGES`). If that fails you can select a different language by giving this option a two-letter language code.

--machine-readable

In lieu of the normal output intended for human consumption, write to standard output in a fixed format more suitable for machine parsing. See the “MACHINE-READABLE OUTPUT” section below for details.

--package-logfile *file*

tlmgr logs all package actions (install, remove, update, failed updates, failed restores) to a separate log file, by default `TEXMFSYSVAR/web2c/tlmgr.log`. This option allows you to select a different file for that. This is separate from normal logging; for that, see the option `-v` below, and `TeXLive::TLUtils`.

--pause

This option make **tlmgr** wait for user input before exiting. Useful on Windows to avoid command windows disappearing.

-debug-translation

This switch will make `tlmgr` report any missing or probably untranslated messages in GUI mode to standard error.

The standard options for TeX Live programs are also accepted: `--help/-h/-?`, `--version`, `-q` (no informational messages), `-v` (debugging messages, can be repeated). For the details about these, see the `TeXLive::TLUtils` documentation.

The `--version` option shows version information about the TeX Live release and about the `tlmgr` script itself. If paired with `-v`, revision number for the used TeX Live Perl modules are shown, too.

ACTIONS**help**

Gives this help information (same as `--help`).

version

Gives version information (same as `--version`).

If `-v` has been given the revisions of the used modules are reported, too.

gui

Start the graphical user interface.

install [option]... pkg...

Install each *pkg* given on the command line. By default this installs all packages on which the given *pkgs* are dependent, also. Options:

--reinstall

Reinstall a package (including dependencies for collections) even if it seems to be already installed (i.e, is present in the TLPDB). This is useful to recover from accidental removal of files in the hierarchy.

When re-installing, only dependencies on normal packages are followed (not those of category Scheme or Collection).

--no-depends

Do not install dependencies. (By default, installing a package ensures that all dependencies of this package are fulfilled.)

--no-depends-at-all

When you install a package which ships binary files the respective binary package will also be installed. That is, for a package `foo`, the package `foo.i386-linux` will also be installed on an `i386-linux` system. This switch suppresses this behavior, and also implies `--no-depends`. Don't use it unless you are sure of what you are doing.

--dry-run

Nothing is actually installed; instead, the actions to be performed are written to the terminal.

--force

If updates to `tlmgr` itself (or other parts of the basic infrastructure) are present, `tlmgr` will bail out and not perform the installation unless this option is given. Not recommended.

update [*option*]... [*pkg*]...

Updates the packages given as arguments to the latest version available at the installation source. Either `--all` or at least one *pkg* name must be specified. Options:

--all

Update all installed packages except for `tlmgr` itself. Thus, if updates to `tlmgr` itself are present, this will simply give an error, unless also the option `--force` or `--self` is given. (See below.)

In addition to updating the installed packages, during the update of a collection the local installation is (by default) synchronized to the status of the collection on the server, for both additions and removals.

Specifically, if a package has been removed on the server (and thus has also been removed from the respective collection), `tlmgr` will remove the package also on the local installation. This is called “auto-remove” and is announced as such when using the option `--list`. This auto-removal can be suppressed using the option `--no-auto-remove`.

Analogously, if a package has been added to an installed collection on the server, it will be added to the local installation. This is called “auto-install” and is announced as such when using the option `--list`. This auto-installation can be suppressed using the option `--no-auto-install`.

An exception to the collection dependency checks (including the auto-installation of packages just mentioned) are those that have been forcibly removed by you (see the **remove** action, `--force` option). To reinstall any such forcibly removed packages use `--rein-stall-forcibly-removed`.

If you want to exclude some packages from being updated (e.g., due to a slow link), see the `--exclude` option below.

--self

Update `tlmgr` itself (that is, the infrastructure packages) if updates to it are present. On Windows this includes updates to the private Perl interpreter shipped inside TeX Live.

If this option is given together with either `--all` or a list of packages, then `tlmgr` will be updated first and, if this update succeeds, the new version will be restarted to complete the rest of the updates.

In short:

```
tlmgr update --self          # update infrastructure only
tlmgr update --self --all    # update infrastructure and all packages
tlmgr update --force --all   # update all packages but *not* infrastructure
```

--list [*pkg*]

Concisely list the packages which would be updated, newly installed, or removed, without actually changing anything. If no package is given acts like `--list --all`, otherwise lists only updates to the packages given as arguments (and its dependencies).

--exclude *pkg*

Exclude *pkg* from the update process. If this option is given more than once, its arguments accumulate.

An argument *pkg* excludes both the package *pkg* itself and all its related architecture-specific packages *pkg.ARCH*. For example,

```
tlmgr update --all --exclude a2ping
```

will not update *a2ping*, *a2ping.i386-linux*, or any other *a2ping.ARCH* package.

If this option specifies a package that would otherwise be a candidate for auto-installation, auto-removal, or reinstallation of a forcibly removed package, *tlmgr* quits with an error message. Excludes are not supported in these circumstances.

--dry-run

Nothing is actually installed; instead, the actions to be performed are written to the terminal. This is a more detailed report than `--list`.

--force

Force update of normal packages, without updating *tlmgr* itself (unless the `--self` option is also given). Not recommended.

Also, `update --list` is still performed regardless of this option.

--no-auto-remove

Under normal circumstances *tlmgr* tries to remove packages which have disappeared on the server when called with `--all`, as described above. This prevents any such removals.

--no-auto-install

Under normal circumstances *tlmgr* will install packages which are new on the server when called with `--all`. This prevents any such automatic installation. Note that after this the packages that would have been auto-installed will be considered as forcibly removed. Actually, if ‘foobar’ is the only new package on server, then

```
tlmgr update --all --no-auto-install
```

is equivalent to

```
tlmgr update --all
tlmgr remove --force foobar
```

--reinstall-forcibly-removed

Under normal circumstances *tlmgr* will not install packages that have been forcibly removed by the user; that is, removed with `remove --force`, or whose installation was prohibited by `--no-auto-install` during an earlier update.

This option makes *tlmgr* ignore the forcible removals and re-install all such packages. This can be used to synchronize an installation with the server’s idea of what is available by calling

```
tlmgr update --reinstall-forcibly-removed --all
```

This option only works in conjunction with `--all`.

--backup and **--backupdir** *directory*

These two options control the creation of backups of packages *before* updating; that is, backup of packages as currently installed. If neither of these options are given, no backup package will be saved. If **--backupdir** is given and specifies a writable directory then a backup will be made in that location. If only **--backup** is given, then a backup will be made to the directory previously set via the `option` action (see below). If both are given then a backup will be made to the specified *directory*.

You can set options via the `option` action to automatically create backups for all packages, and/or keep only a certain number of backups. Please see the `option` action for details.

`tlmgr` always makes a temporary backup when updating packages, in case of download or other failure during an update. In contrast, the purpose of this **--backup** option is to allow you to save a persistent backup in case the actual *content* of the update causes problems, e.g., introduces an incompatibility.

The `restore` action explains how to restore from a backup.

--no-depends

If you call for updating a package normally all depending packages will also be checked for updates and updated if necessary. This switch suppresses this behavior.

--no-depends-at-all

See above under **install** (and beware).

If the package on the server is older than the package already installed (e.g., if the selected mirror is out of date), `tlmgr` does not downgrade. Also, packages for uninstalled platforms are not installed.

backup [**--clean**[=*N*]] [**--backupdir** *dir*] [**--all**] [*pkg*]...

If the **--clean** option is not specified, this action makes a backup of the given packages, or all packages given **--all**. These backups are saved to the value of **--backupdir** option if that is an existing and writable directory; if **--backupdir** is not given, the `backupdir` option setting in the TLPDB is used, if present. If both are missing, no backups are made.

If the **--clean** option is specified, backups are cleaned (pruned) instead of made. If the value for the **--clean** switch is not given, the value of the `autobackup` option is used. If both are missing, an error is issued. For the details of backup cleaning, see the `option` action below.

Options:

--backupdir *directory*

The *directory* argument is required and must specify an existing directory where backups are to be placed.

--all

Make a backup of all packages in the TeX Live installation. This will take quite a lot of space and time.

--clean[=*N*]

Instead of making backups, prune the backup directory of old backups, as explained above.

--dry-run

Nothing is actually backed up or removed; instead, the actions to be performed are written to the terminal.

restore **--backupdir** *dir* [*pkg* [*rev*]

Restore a package from a previously-made backup.

If neither *pkg* nor *rev* are given, list the available backup revisions for all packages.

With *pkg* given but no *rev*, list all available backup revisions of *pkg*.

With both *pkg* and *rev*, tries to restore the package from the specified backup.

Options:

--dry-run

Nothing is actually restored; instead, the actions to be performed are written to the terminal.

--backupdir *directory*

Specify the directory where the backups are to be found.

remove [*option*]... *pkg*...

Remove each *pkg* specified. Removing a collection removes all package dependencies (unless **--no-depends** is specified), but not any collection dependencies of that collection. However, when removing a package, dependencies are never removed. Options:

--no-depends

Do not remove dependent packages.

--no-depends-at-all

See above under **install** (and beware).

--force

By default, removal of a package or collection that is a dependency of another collection or scheme is not allowed. With this option, the package will be removed unconditionally. Use with care.

A package that has been removed using the **--force** option because it is still listed in an installed collection or scheme will not be updated, and will be mentioned as **forcibly removed** in the output of **tlmgr update --list**.

--dry-run

Nothing is actually removed; instead, the actions to be performed are written to the terminal.

option

option [*show*]

option *showall*

option *key* [*value*]

The first form shows the global TeX Live settings currently saved in the TLPDB with a short description and the key used for changing it in paranthesis.

The second form acts like the first, but also shows options which can be defined but are not currently set to any value.

In the third form, if *value* is not given, the setting for *key* is displayed. If *value* is present, *key* is set to *value*.

Possible values for *key* are (but see **tlmgr option showall** for the definitive list):

```

repository (default package repository),
formats    (create formats at installation time),
docfiles   (install documentation files),
srcfiles   (install source files),
backupdir  (default directory for backups),
autobackup (number of backups to keep).
sys_bin    (directory to which executables are linked by the path action)
sys_man    (directory to which man pages are linked by the path action)
sys_info   (directory to which Info files are linked by the path action)

```

One common use of option is to permanently change the installation to get further updates from the Internet, after originally installing from DVD. To do this, you can run

```
tlmgr option repository http://mirror.ctan.org/systems/texlive/tlnet
```

The `install-tl` documentation has more information about the possible values for `repository`.

To keep backward compatibility `location` can be used as alternative name for `repository`.

The two options `autobackup` and `backupdir` determine defaults for the `update`, `backup` and `restore` actions. These three actions need a directory to write to or read from the backups. If `--backupdir` is not specified on the command line, the `backupdir` option value is used (if set).

The `autobackup` option (de)activates automatic generation of backups. Its value is an integer. If the `autobackup` value is `-1`, no backups are removed. If `autobackup` is `0` or more, it specifies the number of backups to keep. Thus, backups are disabled if the value is `0`. In the `--clean` mode of the `backup` action this option also specifies the number to be kept.

To setup `autobackup` to `-1` on the command line, use

```
tlmgr option autobackup infity
```

Or you can use:

```
tlmgr option -- autobackup -1
```

The `--` avoids having the `-1` treated as an option. (For most programs, `--` stops parsing for arguments at the point where it appears; this is a general feature of option parsing.)

conf

Show general configuration information for TeX Live, including active configuration files, path settings, and more.

This option to `tlmgr` mimics the `texconfig conf` call, but works on all supported platforms.

path [--w32mode=user|admin] [add|remove]

On Unix adds or removes symlinks for binaries, man pages, and info pages in the directories specified by the respective options (see above).

On Windows, the registry part where the binary directory is added or removed is determined in the following way:

If the user has admin rights, and the option `--w32mode` is not given, the setting `w32_multi_user` determines the location (i.e., if it is on then the system path, otherwise the user path is changed).

If the user has admin rights, and the option `--w32mode` is given, this option determines the path to be adjusted.

If the user does not have admin rights, and the option `--w32mode` is not given, and the setting `w32_multi_user` is off, the user path is changed, while if the setting `w32_multi_user` is on, a warning is issued that the caller does not have enough privileges.

If the user does not have admin rights, and the option `--w32mode` is given, it must be **user** and the user path will be adjusted. If a user without admin rights uses the option `--w32mode admin` a warning is issued that the caller does not have enough privileges.

postaction [`--w32mode=user|admin`] [`--fileassocmode=1|2`] [`-all`] [`install|remove`] [`shortcut|fileassoc|script`] [`<package> <package> ...`]

Carry out the `postaction shortcut`, `fileassoc`, or `script` as given as second obligatory argument in `install` or `remove` mode (first obligatory argument), for either the packages given on the command line, or for all if `-all` is given.

The option `--w32mode` is `user` all actions will only be carried out in the user accessible parts of the registry/filesystem, while the `admin` mode selects the system wide parts of the registry for the file associations. Note that if you do not have enough permissions using `--w32mode=admin` will not succeed.

For the `postaction fileassoc` the mode can be set with `--fileassocmode`. If it is set to 1, only new associations are added, if it is set to 2, all associations are set to the TeX Live programs.

paper

paper [`a4|letter`]

xdvi | **pdftex** | **dvips** | **dvipdfmx** | **dvipdfm** | **context paper** [`papersize`] [`--list`]

With no arguments (`tlmgr paper`), shows the default paper size setting for all known programs.

With one argument (e.g., `tlmgr paper a4`), sets the default for all known programs to that paper size.

With a program given as the first argument and no paper size specified (e.g., `tlmgr dvips paper`), shows the default paper size for that program.

With a program given as the first argument and a paper size as the last argument (e.g., `tlmgr dvips paper a4`), set the default for that program to that paper size.

With a program given as the first argument and `--list` given as the last argument (e.g., `tlmgr dvips paper --list`), shows all valid paper sizes for that program. The first size shown is the default.

Incidentally, this syntax of having a specific program name before the `paper` keyword may seem strange. It is inherited from the longstanding `texconfig` script, which supports other configuration settings for some programs, notably `dvips`. `tlmgr` does not support those extra settings at present.

arch list|add|remove *arch*...

arch list lists the TeX Live names of all the architectures (*i386-linux*, ...) available at the package repository.

arch add *arch*... adds the executables for each given architecture *arch* to the installation.

arch remove *arch*... removes the executables for each given architecture *arch* from the installation, but keeps the currently running architecture in any case.

Options:

--dry-run

Nothing is actually installed; instead, the actions to be performed are written to the terminal.

print-arch

Prints the currently detected architecture name to standard output.

search [*option*]... *what*

By default searches the names, short and long descriptions of all locally installed packages for the given argument (interpreted as regexp). Options:

--file

List all filenames containing *what*.

--global

Search the TeX Live Database of the installation medium, instead of the local installation.

show [--list] *pkg*...

Shows information about *pkg*: the name, category, installation status, short and long description. Searches in the remote installation source for the package if it is not locally installed.

If the option **--list** is given with a package, the list of contained files is also shown, including those for architecture specific dependencies. When given with schemes and collections, **--list** outputs their dependencies in a similar way.

list [--only-installed] [*collections*|*schemes*|*pkg*...]

With no argument, lists all packages available at the package repository, prefixing those already installed with *i*.

With the single word *collections* or *schemes* as the argument, lists the request type.

With anything else as arguments, operates as the **show** action.

If the option **--only-installed** is given the installation source will not be used and only locally installed packages, collections, or schemes are listed.

check [*option*]... [*files*|*depends*|*executes*|*runfiles*|*all*]

Executes one (or all) check(s) on the consistency of the installation.

files

Checks that all files listed in the TeX Live Database (*texlive.tlpdb*) are actually present, and lists those missing.

depends

Lists those packages which occur as dependencies in an installed collections, but are themselves not installed, and those packages that are not contained in any collection.

If you call `tlmgr check collections` this test will be carried out instead since former versions for `tlmgr` called it that way.

executes

Check that the files referred to by `execute` directives in the TeX Live Database are present.

runfiles

List those filenames that are occurring more than one time in the runfiles.

Options:

--use-svn

Use the output of `svn status` instead of listing the files; for checking the TL development repository.

uninstall

Uninstalls the entire TeX Live installation. Options:

--force

Do not ask for confirmation, remove immediately.

generate [*option*]... *what*

generate language

generate language.dat

generate language.def

generate fmtutil

generate updmap

The `generate` action overwrites any manual changes made in the respective files: it recreates them from scratch.

For `fmtutil` and the language files, this is normal, and both the TeX Live installer and `tlmgr` routinely call *generate* for them.

For `updmap`, however, neither the installer nor `tlmgr` use `generate`, because the result would be to disable all maps which have been manually installed via `updmap-sys --enable`, e.g., for proprietary or local fonts. The *generate* action only incorporates the changes in the `--localcfg` file mentioned below. Furthermore, `tlmgr` updates and maintains the final `updmap.cfg` in `TEXMFSYSCONFIG` (while the other files are in `TEXMFSYSVAR`), because that is the location that `updmap-sys` (via `tcfmgr`) uses.

Notwithstanding the above, if you only use the fonts and font packages within TeX Live, and maintain your local fonts (if any) using `updmap-local.cfg`, there is nothing wrong with using `generate updmap`. Indeed, we use it ourselves to generate the `updmap.cfg` file in the live source repository.

In more detail: `generate` remakes any of the four config files `language.dat`, `language.def`, `fmtutil.cnf`, and `updmap.cfg` from the information present in the local TLPDB, plus locally-maintained files.

The locally-maintained files are `language-local.dat`, `language-local.def`, `fmtutil-local.cnf`, or `updmap-local.cfg`, searched for in `TEXMFLOCAL` in the respective directories. If they are present, the final file is made by starting with the main file, omitting any entries that the local file specifies to be disabled, and finally appending the local file.

Local files specify entries to be disabled with a comment line like this:

```
#!NAME
%!NAME
```

where `fmtutil.cnf` and `updmap.cfg` use `#`, and `language.dat` and `language.def` use `%`. In any case, the *name* is the respective format name, map file name (include the `.map` extension), or hyphenation pattern identifier. Examples:

```
#!pdflatex
#!lm.map
%!german
```

(Of course, you're not likely to actually want to disable those particular items. They're just examples.)

After such a disabling line, the local file can include another entry for the same item, if a different definition is desired.

The form `generate language` recreates both the `language.dat` and the `language.def` files, while the forms with extension recreates only the given language file.

Options:

--dest *output file*

specifies the output file (defaults to the respective location in `TEXMFSYSVAR` for `language*` and `fmtutil`, and `TEXMFCONFIG` for `updmap`). If `--dest` is given to `generate language`, its value will be used for the `language.dat` output, and `.def` will be appended to the value for the name of the `language.def` output file. (This is just to avoid overwriting; if you want a specific name for each output file, we recommend invoking `tlmgr` twice.)

--localcfg *local conf file*

specifies the (optional) local additions (defaults to the respective location in `TEXMFLOCAL`).

The respective locations are as follows:

```
tex/generic/config/language.dat (and language-local.dat);
tex/generic/config/language.def (and language-local.def);
web2c/fmtutil.cnf (and fmtutil-local.cnf);
web2c/updmap.cfg (and updmap-local.cfg).
```

Final repetition: as explained above, `tlmgr` does *not* use the equivalent of `generate updmap` for map files. Therefore, if you want to make use of `updmap-local.cfg`, you need to run `tlmgr generate updmap` yourself. After that, network updates and your local changes should be merged.

MACHINE-READABLE OUTPUT

Given the `--machine-readable` option, `tlmgr` writes to `stdout` in the fixed line-oriented format described here, and the usual informational messages for human consumption are written to `stderr` (normally they are written to `stdout`). The idea is that a program can get all the information it needs by reading `stdout`.

Currently this option only applies to the update, the install, and the option actions.

update and install actions

The output format is as follows:

```
fieldname "\t" value
...
"end-of-header"
pkgname status localrev serverrev size estrem esttot
...
"end-of-updates"
other output from post actions, not in machine readable form
```

The header section currently has two fields: `location-url` (the repository source from which updates are being drawn), and `total-bytes` (the total number of bytes to be downloaded).

The `localrev` and `serverrev` fields for each package are the revision numbers in the local installation and server repository, respectively. The `size` field is the number of bytes to be downloaded, i.e., the size of the compressed tar file for a network installation, not the unpacked size. The `estrem` and `esttot` fields are only present for updated and auto-install packages, and contain the estimated remaining time and the estimated total time.

Line endings may be either LF or CRLF depending on the current platform.

`location-url` *location*

The *location* may be a url (including `file:///foo/bar/...`), or a directory name (`/foo/bar`). It is the package repository from which the new package information was drawn.

`total-bytes` *count*

The *count* is simply a decimal number, the sum of the sizes of all the packages that need updating or installing (which are listed subsequently).

Then comes a line with only the literal string `end-of-header`.

Each following line until a line with literal string `end-of-updates` reports on one package. The fields on each line are separated by a tab. Here are the fields.

pkgname

The TeX Live package identifier, with a possible architecture suffix for executables. For instance, `pdftex` and `pdftex.i386-linux` are given as two separate packages, one on each line.

status

The status of the package update. One character, as follows:

- d The package was removed on the server.
- f The package was removed in the local installation, even though a collection depended on it. (E.g., the user ran `tlmgr remove --force`.)
- u Normal update is needed.
- r Reversed non-update: the locally-installed version is newer than the version on the server.

- a Automatically-determined need for installation, the package is new on the server and is (most probably) part of an installed collection.
- i Package will be installed and isn't present in the local installation (action install).
- I Package is already present but will be reinstalled (action install).

localrev

The revision number of the installed package, or – if it is not present locally.

serverrev

The revision number of the package on the server, or – if it is not present on the server.

size

The size in bytes of the package on the server. The sum of all the package sizes is given in the `total-bytes` header field mentioned above.

estrem

The estimated remaining time.

esttot

The estimated total time.

option action:

The output format is as follows:

```
key "\t" value
```

If a value is not saved in the database the string `(not set)` is shown.

If you are developing a program that uses this output, and find that changes would be helpful, do not hesitate to write the mailing list.

AUTHORS AND COPYRIGHT

This script and its documentation were written for the TeX Live distribution (<http://tug.org/texlive>) and both are licensed under the GNU General Public License Version 2 or later.