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. For information on initially downloading and installing TeX Live, see <http://tug.org/texlive/acquire.html>.

The most up-to-date version of this documentation (updated nightly from the development sources) is available at <http://tug.org/texlive/tlmgr.html>, along with procedures for updating `tlmgr` itself and information about test versions.

TeX Live is organized into a few top-level *schemes*, each of which is specified as a different set of *collections* and *packages*, where a collection is a set of packages, and a package is what contains actual files. Schemes typically contain a mix of collections and packages, but each package is included in exactly one collection, no more and no less. A TeX Live installation can be customized and managed at any level.

See <http://tug.org/texlive/doc> for all the TeX Live documentation available.

EXAMPLES

After successfully installing TeX Live, here are a few common operations with `tlmgr`:

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

Tell `tlmgr` to use a nearby CTAN mirror for future updates; useful if you installed TeX Live from the DVD image and want continuing updates.

`tlmgr update --list`

Report what would be updated without actually updating anything.

`tlmgr update --all`

Make your local TeX installation correspond to what is in the package repository (typically useful when updating from CTAN).

`tlmgr info` *pkg*

Display detailed information about *pkg*, such as the installation status and description.

For all the capabilities and details of `tlmgr`, please read the following voluminous information.

OPTIONS

The following options to `tlmgr` are global options, not specific to any action. All options, whether global or action-specific, can be given anywhere on the command line, and in any 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 (a.k.a. the TLPDB, defined entirely in the file `tlpkg/texlive.tlpdb`). The documentation for `install-tl` has more details about this (http://tug.org/texlive/doc/install-tl.html <http://tug.org/texlive/doc/install-tl.html>).

`--repository` changes the repository location only for the current run; to make a permanent change, use `option repository` (see the option action).

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

--gui [*action*]
`tlmgr` has a graphical interface as well as the command-line interface. You can give this option, `--gui`, 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. Without any action, the GUI will be started at the main screen.

--gui-lang *llcode*
By default, 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 with a language code (based on ISO 639-1). Currently supported (but not necessarily completely translated) are: English (en, default), Czech (cs), German (de), French (fr), Italian (it), Japanese (ja), Dutch (nl), Polish (pl), Brazilian Portuguese (pt_BR), Russian (ru), Slovak (sk), Slovenian (sl), Serbian (sr), Vietnamese (vi), simplified Chinese (zh_CN), and traditional Chinese (zh_TW).

--machine-readable
Instead of the normal output intended for human consumption, write (to standard output) a fixed format more suitable for machine parsing. See the ''MACHINE-READABLE OUTPUT'' section below.

--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 specific a different file for the log.

--pause
This option makes `tlmgr` wait for user input before exiting. Useful on Windows to avoid disappearing command windows.

--persistent-downloads
--no-persistent-downloads
For network-based installations, this option (on by default) makes `tlmgr` try to set up a persistent connection (using the Net::LWP Perl module). The idea is to open and reuse only one connection per session between your computer and the server, instead of initiating a new download for each package.

If this is not possible, `tlmgr` will fall back to using `wget`. To disable these persistent connections, use `--no-persistent-downloads`.

--no-execute-actions
Suppress the execution of the execute actions as defined in the tlpsrc files. Documented only for completeness, as this is only useful in debugging.

--debug-translation

In GUI mode, this switch makes `tlmgr` report any missing, or more likely untranslated, messages to standard error. Helpful for translators to see what remains to be done.

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 `-v` is given as well, 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. See **GUI** below.

install [*option***]...** ***pkg*****...**

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

--file

Instead of fetching a package from the installation repository, use the packages files given on the command line. These files need to be proper TeX Live package files (with contained tlpobj file).

--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.

This means that if a package has been removed on the server (and thus has also been removed from the respective collection), `tlmgr` will remove the package in 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 a collection on the server that is also installed locally, 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, that is, you called `tlmgr remove --force` on them. (See the `remove` action documentation.) To reinstall any such forcibly removed packages use `--reinstall-forcibly-removed`.

If you want to exclude some packages from the current update run (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* infrastructu
                           # ... this last at your own risk, not recomm
```

--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`.

--list [*pkg*]
Concisely list the packages which would be updated, newly installed, or removed, without actually changing anything. If `--all` is also given, all available updates are listed. If `--self` is given, but not `--all`, only updates to the critical packages (tlmgr, texlive infrastructure, perl on Windows, etc.) are listed. If neither `--all` nor `--self` is given, and in addition no *pkg* is given, then `--all` is assumed (thus, `tlmgr update --list` is the same as `tlmgr update --list --all`). If neither `--all` nor `--self` is given, but specific package names are given, those packages are checked for updates.

--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 platform-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.

--no-auto-remove [*pkg*]...
Under normal circumstances `tlmgr` tries to remove packages which have disappeared on the server, as described above under `--all`. This option prevents any such removals, either for all packages (with `--all`), or the given *pkg* names.

--no-auto-install [*pkg*]...
Under normal circumstances `tlmgr` will install packages which are new on the server, as described above under `--all`. This option prevents any such automatic installation, either for all packages (with `--all`), or the given *pkg* names.

Furthermore, after the `tlmgr` run using this has finished, the packages that would have been auto-installed *will be considered as forcibly removed*. So, if `foobar` is the only new package on the 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 completely synchronize an installation with the server's idea of what is available:

```
tlmgr update --reinstall-forcibly-removed --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).

`--`**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.

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 the `--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 pruned (removed) instead of saved. The optional integer value *N* may be specified to set the number of backups that will be retained when cleaning. If `N` is not given, the value of the `autobackup` option is used. If both are missing, an error is issued. For more details of backup pruning, see the `option` action.

Options:

`--`**backupdir** *directory*

Overrides the `backupdir` option setting in the TLPDB. The *directory* argument is required and must specify an existing, writable directory where backups are to be placed.

`--`**all**

If `--clean` is not specified, make a backup of all packages in the TeX Live installation; this will take quite a lot of space and time. If `--clean` is specified, all packages are pruned.

`--`**clean**[*=N*]
Instead of making backups, prune the backup directory of old backups, as explained above. The optional integer argument *N* overrides the `autobackup` option set in the TLPDB. You must use `--all` or a list of packages together with this option, as desired.

`--`**dry-run**
Nothing is actually backed up or removed; instead, the actions to be performed are written to the terminal.

restore [`--`**backupdir** *dir***] [**`--`**all** | *pkg* **[***rev***]]**
Restore a package from a previously-made backup.

If `--all` is given, try to restore the latest revision of all package backups found in the backup directory.

Otherwise, 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*.

When listing available packages tlmgr shows the revision and in parenthesis the creation time if available (in format yyyy-mm-dd hh:mm).

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

Options:

`--`**all**
Try to restore the latest revision of all package backups found in the backup directory. Additional non-option arguments (like *pkg*) are not allowed.

`--`**backupdir** *directory*
Specify the directory where the backups are to be found. If not given it will be taken from the configuration setting in the TLPDB.

`--`**dry-run**
Nothing is actually restored; instead, the actions to be performed are written to the terminal.

`--`**force**
Don't ask questions.

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.

repository

repository list
repository add *path* [*tag*]
repository remove *path*|*tag*
repository set *path*[#*tag*] [*path*[#*tag*] ...]
This action manages the list of repositories. See ''MULTIPLE REPOSITORIES'' below for detailed explanations.

The first form (`list`) lists all configured repositories and the respective tags if set. The second form (`add`) adds a repository (optionally attaching a tag) to the list of repositories. The third form (`remove`) removes a repository, either by full path/url, or by tag. The last form (`set`) sets the list of repositories to the items given on the command line, not keeping previous settings

In all cases, one of the repositories must be tagged as `main`; otherwise, all operations will fail!

candidates

candidates *pkg*
Shows the available candidate repositories for package *pkg*. See ''MULTIPLE REPOSITORIES'' below.

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 parentheses.

The second form is similar, 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 (run `tlmgr option showall` for the definitive list):

```
repository (default package repository),
formats    (create formats at installation time),
postcode   (run postinst code blobs)
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)
desktop_integration (Windows-only: create Start menu shortcuts)
fileassocs (Windows-only: change file associations)
```

```
multiuser  (Windows-only: install for all users)
```

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`. (For backward compatibility, `location` can be used as alternative name for `repository`.)

If `formats` is set (this is the default), then formats are regenerated when either the engine or the format files have changed. Disable this only when you know what you are doing.

The `postcode` option controls execution of per-package postinstallation action code. It is set by default, and again disabling is not likely to be of interest except perhaps to developers.

The `docfiles` and `srcfiles` options control the installation of their respective files of a package. By default both are enabled (1). This can be disabled (set to 0) if disk space is (very) limited.

The options `autobackup` and `backupdir` determine the defaults for the actions `update`, `backup` and `restore`. These three actions need a directory in which to read or write 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 either:

```
tlmgr option autobackup infty
```

or:

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

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

The `sys_bin`, `sys_man`, and `sys_info` options are used on Unix-like systems to control the generation of links for executables, info files and man pages. See the `path` action for details.

The last three options control behaviour on Windows installations. If `desktop_integration` is set, then some packages will install items in a sub-folder of the Start menu for `tlmgr gui`, documentation, etc. If `fileassocs` is set, Windows file associations are made (see also the `postaction` action). Finally, if `multiuser` is set, then adaptions to the registry and the menus are done for all users on the system instead of only the current user. All three options are on by default.

conf [texmf|tlmgr [*key* **[***value***]]]**

With only `conf`, show general configuration information for TeX Live, including active configuration files, path settings, and more. This is like the `texconfig conf` call, but works on all supported platforms.

With either `conf texmf` or `conf tlmgr` given in addition, shows all key/value pairs (i.e., all

settings) as saved in `ROOT/texmf.cnf` or the tlmgr configuration file (see below), respectively.

If *key* is given in addition, shows the value of only that given *key* in the respective file.

If *value* is given in addition, *key* is set to *value* in the respective file. *No error checking is done!*

Practical application: if the execution of (some or all) system commands via `\write18` was left enabled during installation, you can disable it afterwards:

```
tlmgr conf texmf shell_escape 0
```

Warning: The general facility is here, but tinkering with settings in this way is very strongly discouraged. Again, no error checking is done, so any sort of breakage is possible.

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.

platform list|add|remove *platform*...

`platform list` lists the TeX Live names of all the platforms (a.k.a. architectures), (`i386−linux`, ...) available at the package repository.

`platform add` *platform*... adds the executables for each given platform *platform* to the installation from the repository.

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

`arch` is a synonym for `platform`.

Options:

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

print-platform

Print the TeX Live identifier for the detected platform (hardware/operating system) combination to standard output, and exit. `−−print−arch` is a synonym.

info [*option***...] [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 instead of all packages.

With any other arguments, display information about *pkg*: the name, category, short and long description, installation status, and TeX Live revision number. If *pkg* is not locally installed, searches in the remote installation source.

It also displays information taken from the TeX Catalogue, namely the package version, date, and license. Consider these, especially the package version, as approximations only, due to timing skew of the updates of the different pieces. By contrast, the `revision` value comes directly from TL and is reliable.

The former actions `show` and `list` are merged into this action, but are still supported for backward compatibility.

Options:

--list

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

--only-installed

If this options is given, the installation source will not be used; only locally installed packages, collections, or schemes are listed. (Does not work for listing of packages for now)

--taxonomy
--keyword
--functionality
--characterization

In addition to the normal data displayed, also display information for given packages from the corresponding taxonomy (or all of them). See ''TAXONOMIES'' below for details.

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

search [option...] --file what

search [option...] --taxonomy what

search [option...] --keyword what

search [option...] --functionality what

search [option...] --characterization what

search [option...] --all what

By default, search the names, short descriptions, and long descriptions of all locally installed packages for the argument *what*, interpreted as a regular expression.

Options:

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

--word
Restrict the search to match only full words. For example, searching for `table` with this option will not output packages containing the word `tables` (unless they also contain the word `table` on its own).

--list
If a search for any (or all) taxonomies is done, by specifying one of the taxonomy options below, then instead of searching for packages, list the entire corresponding taxonomy (or all of them). See ''TAXONOMIES'' below.

Other search options are selected by specifying one of the following:

--file
List all filenames containing *what*.

--taxonomy
--keyword
--functionality
--characterization
Search in the corresponding taxonomy (or all) instead of the package descriptions. See ''TAXONOMIES'' below.

--all
Search for package names, descriptions, and taxonomies, but not files.

dump-tlpdb [--local|--remote]
Dump complete local or remote TLPDB to standard output, as-is. The output is analogous to the `--machine-readable` output; see ''MACHINE-READABLE OUTPUT'' section.

Options:

--local
Dump the local tlpdb.

--remote
Dump the remote tlpdb.

Exactly one of `--local` and `--remote` must be given.

In either case, the first line of the output specifies the repository location, in this format:

```
"location-url" "\t" location
```

where `location-url` is the literal field name, followed by a tab, and *location* is the file or url to the repository.

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

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 local TLPDB (`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.

path [--w32mode=user|admin] [add|remove]
On Unix, merely adds or removes symlinks for binaries, man pages, and info pages in the system directories specified by the respective options (see the option description above). Does not change any initialization files, either system or personal.

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

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

`--fileassocmode` specifies the action for file associations. If it is set to 1 (the default), only new associations are added; if it is set to 2, all associations are set to the TeX Live programs. (See also `option fileassocs`.)

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 language.dat.lua
generate fmtutil

The `generate` action overwrites any manual changes made in the respective files: it recreates them from scratch based on the information of the installed packages, plus local adaptions. The TeX Live installer and `tlmgr` routinely call `generate` for all of these files.

For managing your own fonts, please read the `updmap --help` information and/or <http://tug.org/fonts/fontinstall.html>.

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

The locally-maintained files are `language-local.dat`, `language-local.def`, `language-local.dat.lua`, or `fmtutil-local.cnf`, searched for in `TEXMFLOCAL` in the respective directories. If local additions 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.

(Historical note: The formerly supported `updmap-local.cfg` is no longer read, since `updmap` now supports multiple `updmap.cfg` files. Thus, local additions can and should be put into an `updmap.cfg` file in `TEXMFLOCAL`. The `generate updmap` action no longer exists.)

Local files specify entries to be disabled with a comment line, namely one of these:

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

where `fmtutil.cnf` uses `#`, `language.dat` and `language.def` use `%`, and `language.dat.lua` use `--`. In all cases, the *name* is the respective format name or hyphenation pattern identifier. Examples:

```
#!pdflatex
%!german
--!usenglishmax
```

(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. In general, except for the special disabling lines, the local files follow the same syntax as the master files.

The form `generate language` recreates all three files `language.dat`, `language.def`,

and `language.dat.lua`, while the forms with an extension recreates only that given language file.

Options:

`--`**dest** *output_file*
: specifies the output file (defaults to the respective location in `TEXMFSYSVAR`). If `--dest` is given to `generate language`, it serves as a basename onto which `.dat` will be appended for the name of the `language.dat` output file, `.def` will be appended to the value for the name of the `language.def` output file, and `.dat.lua` to the name of the `language.dat.lua` 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`).

`--`**rebuild-sys**
: tells tlmgr to run necessary programs after config files have been regenerated. These are: `fmtutil-sys --all` after `generate fmtutil`, `fmtutil-sys --byhyphen .../language.dat` after `generate language.dat`, and `fmtutil-sys --byhyphen .../language.def` after `generate language.def`.

  These subsequent calls cause the newly-generated files to actually take effect. This is not done by default since those calls are lengthy processes and one might want to made several related changes in succession before invoking these programs.

The respective locations are as follows:

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

TLMGR CONFIGURATION FILE

A small subset of the command line options can be set in a config file for `tlmgr` which resides in `TEXMFCONFIG/tlmgr/config`. By default, the config file is in `~/.texliveYYYY/texmf-config/tlmgr/config` (replacing `YYYY` with the year of your TeX Live installation). This is *not* `TEXMFSYSVAR`, so that the file is specific to a single user.

In this file, empty lines and lines starting with # are ignored. All other lines must look like

```
key = value
```

where the allowed keys are `gui-expertmode` (values 0 or 1), `persistent-downloads` (values 0 or 1), `auto-remove` (values 0 or 1), and `gui-lang` (values like the command line arguments). `persistent-downloads`, `gui-lang`, and `auto-remove` correspond to the respective command line options of the same name. `gui-expertmode` switches between the full GUI and a simplified GUI with only the important and mostly used settings.

TAXONOMIES

tlmgr allows searching and listing of various categorizations, which we call *taxonomies*, as provided by an enhanced TeX Catalogue (available for testing at <http://az.ctan.org>). This is useful when, for example, you don't know a specific package name but have an idea of the functionality you need; or when you want to see all packages relating to a given area.

There are three different taxonomies, specified by the following options:

−−keyword
The keywords, as specified at <http://az.ctan.org/keyword>.

−−functionality
The ‘‘by−topic’’ categorization created by J\"urgen Fenn, as specified at http://az.ctan.org/characterization/by−function <http://az.ctan.org/characterization/by-function>.

−−characterization
Both the primary and secondary functionalities, as specified at <http://az.ctan.org/characterization/choose_dimen>.

−−taxonomy
Operate on all the taxonomies.

The taxonomies are updated nightly and stored within TeX Live, so Internet access is not required to search them.

Examples:

```
tlmgr search −−taxonomy exercise      # check all taxonomies for "exercise
tlmgr search −−taxonomy −−word table  # check for "table" on its own
tlmgr search −−list −−keyword         # dump entire keyword taxonomy
tlmgr show −−taxonomy pdftex          # show pdftex package information,
                                      #   including all taxonomy entries
```

MULTIPLE REPOSITORIES

The main TeX Live repository includes a vast array of packages. Nevertheless, additional local repositories can be useful to provide locally-installed resources, such as proprietary fonts and house styles. Also, alternative package repositories distribute packages that cannot or should not be included in TeX Live, due to being under rapid development or for other reasons.

The simplest and most reliable method is simply to temporarily set the installation source to any repository (with the `−repository` command line option or `option repository`), and perform your operations. When you are using multiple repositories over a sustained time, however, this is inconvenient. Thus, it’s possible to tell `tlmgr` about additional repositories you want to use. The basic command is `tlmgr repository add`. The rest of this section explains further.

When using multiple repositories, one of them has to be set as the main repository, which distributes most of the installed packages. If you switch from a single repository installation to a multiple repository installation, the previously set repository will be set as the main repository.

By default, even if multiple repositories are configured, packages are *only* installed from the main repository. Thus, simply adding a second repository does not actually enable installation of anything from there. You also have to specify which packages should be taken from a different repository by specifying so-called ‘‘pinning’’ rules, described next.

Pinning

Pinning a package is done by editing (creating) this file:

```
TEXMFLOCAL/tlpkg/pinning.txt
```

with lines of the form:

```
repo:pkg[,pkg]
```

In this line, the *repo* is either a full url or repository tag that was added to the repository list. Each *pkg* is a shell-style glob for package identifiers.

When a package `foo` is pinned to a repository, a package `foo` in any other repository, even if it has a higher revision number, will not be considered an installable candidate.

By default, everything is pinned to the main repository, as if the last line of `pinning.txt` was

```
main:*
```

Multiple repository example with tlcontrib

First, check that we have support for multiple repositories, and have only one enabled (as is the case by default):

```
$ tlmgr repository list
List of repositories (with tags if set):
  /var/www/norbert/tlnet
```

Let's add the `tlcontrib` repository (<http://tlcontrib.metatex.org>, maintained by Taco Hoekwater et al.), with the tag `tlcontrib`:

```
$ tlmgr repository add http://tlcontrib.metatex.org/2012 tlcontrib
```

Check the repository list again:

```
$ tlmgr repository list
List of repositories (with tags if set):
   http://tlcontrib.metatex.org/2011 (tlcontrib)
   /var/www/norbert/tlnet (main)
```

Specify a pinning entry to get the package `microtype` from `tlcontrib`:

```
$ tlocal=`kpsewhich -var-value TEXMFLOCAL`
$ echo "tlcontrib:microtype" > $tlocal/tlpkg/pinning.txt
```

Check that we can find `microtype`:

```
$ tlmgr show microtype
tlmgr: using pinning file .../tlpkg/pinning.txt
tlmgr: package repositories:
...
package:     microtype
category:    Package
...
```

− install `microtype`:

```
$ tlmgr install microtype
tlmgr: using pinning file .../tlpkg/pinning.txt
tlmgr: package repositories:
...
[1/1,  ??:??/??:??] install: microtype @tlcontrib [39k]
```

In the output here you can see that the `microtype` package is installed from the `tlcontrib` repository (`@tlcontrib`). (By the way, hopefully the new version of microtype that is in tlcontrib as of this writing will be released on CTAN soon, but meanwhile, it serves as an

example.)

GUI FOR TLMGR

The graphical user interface for `tlmgr` needs Perl/Tk to be installed. For Windows the necessary modules are shipped within TeX Live, for all other (i.e., Unix-based) systems Perl/Tk (as well as Perl of course) has to be installed. <http://tug.org/texlive/distro.html#perltk> has a list of invocations for some distros.

When started with `tlmgr gui` the graphical user interface will be shown. The main window contains a menu bar, the main display, and a status area where messages normally shown on the console are displayed.

Within the main display there are three main parts: the `Display configuration` area, the list of packages, and the action buttons.

Also, at the top right the currently loaded repository is shown; this also acts as a button and when clicked will try to load the default repository. To load a different repository, see the `tlmgr` menu item.

Finally, the status area at the bottom of the window gives additional information about what is going on.

Main display

Display configuration area

The first part of the main display allows you to specify (filter) which packages are shown. By default, all are shown. Changes here are reflected right away.

Status
: Select whether to show all packages (the default), only those installed, only those *not* installed, or only those with update available.

Category
: Select which categories are shown: packages, collections, and/or schemes. These are briefly explained in the ‘‘DESCRIPTION’’ section above.

Match
: Select packages matching for a specific pattern. By default, this uses the same algorithm as `tlmgr search`, i.e., searches everything: descriptions, taxonomies, and/or filenames. You can also select any subset for searching.

Selection
: Select packages to those selected, those not selected, or all. Here, ‘‘selected’’ means that the checkbox in the beginning of the line of a package is ticked.

Display configuration buttons
: To the right there are three buttons: select all packages, select none (a.k.a. deselect all), and reset all these filters to the defaults, i.e., show all available.

Package list area

The second are of the main display lists all installed packages. If a repository is loaded, those that are available but not installed are also listed.

Double clicking on a package line pops up an informational window with further details: the long description, included files, etc.

Each line of the package list consists of the following items:

a checkbox
: Used to select particular packages; some of the action buttons (see below) work only on the selected packages.

package name
: The name (identifier) of the package as given in the database.

local revision (and version)
: If the package is installed the TeX Live revision number for the installed package will be shown. If there is a catalogue version given in the database for this package, it will be shown in parentheses. However, the catalogue version, unlike the TL revision, is not guaranteed to reflect what is actually installed.

remote revision (and version)
: If a repository has been loaded the revision of the package in the repository (if present) is shown. As with the local column, if a catalogue version is provided it will be displayed. And also as with the local column, the catalogue version may be stale.

short description
: The short description of the package.

Main display action buttons

Below the list of packages are several buttons:

Update all installed
: This calls `tlmgr update --all`, i.e., tries to update all available packages. Below this button is a toggle to allow reinstallation of previously removed packages as part of this action.

  The other four buttons only work on the selected packages, i.e., those where the checkbox at the beginning of the package line is ticked.

Update
: Update only the selected packages.

Install
: Install the selected packages; acts like `tlmgr install`, i.e., also installs dependencies. Thus, installing a collection installs all its constituent packages.

Remove
: Removes the selected packages; acts like `tlmgr remove`, i.e., it will also remove dependencies of collections (but not dependencies of normal packages).

Backup
: Makes a backup of the selected packages; acts like `tlmgr backup`. This action needs the option `backupdir` set (see `Options` – General>).

Menu bar

The following entries can be found in the menu bar:

tlmgr menu
: The items here load various repositories: the default as specified in the TeX Live database, the default network repository, the repository specified on the command line (if any), and an arbitrarily manually-entered one. Also has the so-necessary `quit` operation.

Options menu

Provides access to several groups of options: `Paper` (configuration of default paper sizes), `Platforms` (only on Unix, configuration of the supported/installed platforms), `GUI Language` (select language used in the GUI interface), and `General` (everything else).

Several toggles are also here. The first is `Expert options`, which is set by default. If you turn this off, the next time you start the GUI a simplified screen will be shown that display only the most important functionality. This setting is saved in the configuration file of `tlmgr`; see ''CONFIGURATION FILE'' for details.

The other toggles are all off by default: for debugging output, to disable the automatic installation of new packages, and to disable the automatic removal of packages deleted from the server. Playing with the choices of what is or isn't installed may lead to an inconsistent TeX Live installation; e.g., when a package is renamed.

Actions menu

Provides access to several actions: update the filename database (aka `ls-R`, `mktexlsr`, `texhash`), rebuild all formats (`fmtutil-sys --all`), update the font map database (`updmap-sys`), restore from a backup of a package, and use of symbolic links in system directories (not on Windows).

The final action is to remove the entire TeX Live installation (also not on Windows).

Help menu

Provides access to the TeX Live manual (also on the web at <http://tug.org/texlive/doc.html>) and the usual ''About'' box.

MACHINE-READABLE OUTPUT

With 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 runtime 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 runtime and esttot fields are only present for updated and auto-install packages, and contain the currently passed time since start of installation/updates 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 platform 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.

runtime
: The run time since start of installations or updates.

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