NAME

tlmgr – the native 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 <https://tug.org/texlive/acquire.html>.

The most up-to-date version of this documentation (updated nightly from the development sources) is available at <https://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 <https://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 ctan`
`tlmgr option repository`
`https://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 to have continuing updates. The two commands are equivalent; `ctan` is just an alias for the given url.

Caveat: `mirror.ctan.org` resolves to many different hosts, and they are not perfectly synchronized; we recommend updating only daily (at most), and not more often. You can choose a particular mirror if problems; the list of all CTAN mirrors with the status of each is at <https://ctan.org/mirrors/mirmon>.

`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` *what*

Display detailed information about a package *what*, such as the installation status and description, of searches for *what* in all packages.

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*

Specify the package repository from which packages should be installed or updated, either a local directory or network location, as below. This overridesthe default package repository found in the installation's TeX Live Package Database (a.k.a. the TLPDB, which is given entirely in the file `tlpkg/texlive.tlpdb`).

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

As an example, you can choose a particular CTAN mirror with something like this:

```
-repository http://ctan.example.org/its/ctan/dir/systems/texlive/tlnet
```

Of course a real hostname and its particular top-level CTAN directory have to be specified. The list of CTAN mirrors is available at <https://ctan.org/mirrors/mirmon>.

Here's an example of using a local directory:

```
-repository /local/TL/repository
```

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

Locations can be specified as any of the following:

`/some/local/dir`
`file:/some/local/dir`
: Equivalent ways of specifying a local directory.

`ctan`
`https://mirror.ctan.org/systems/texlive/tlnet`
: Pick a CTAN mirror automatically, trying for one that is both nearby and up-to-date. The chosen mirror is used for the entire download. The bare `ctan` is merely an alias for the full url. (See <https://ctan.org> for more about CTAN and its mirrors.)

`http://server/path/to/tlnet`
: Standard HTTP. If the (default) LWP method is used, persistent connections are supported. TL can also use `curl` or `wget` to do the downloads, or an arbitrary user-specified program, as described in the `tlmgr` documentation (<https://tug.org/texlive/doc/tlmgr.html#ENVIRONMENT-VARIABLES>).

`https://server/path/to/tlnet`
: Again, if the (default) LWP method is used, this supports persistent connections. Unfortunately, some versions of `wget` and `curl` do not support https, and even when `wget` supports https, certificates may be rejected even when the certificate is fine, due to a lack of local certificate roots. The simplest workaround for this problem is to use http or ftp.

`ftp://server/path/to/tlnet`
: If the (default) LWP method is used, persistent connections are supported.

`user@machine:/path/to/tlnet`
`scp://user@machine/path/to/tlnet`
`ssh://user@machine/path/to/tlnet`
: These forms are equivalent; they all use `scp` to transfer files. Using `ssh-agent` is recommended. (Info: <https://en.wikipedia.org/wiki/OpenSSH>,

<https://en.wikipedia.org/wiki/Ssh-agent>.)

If the repository is on the network, trailing `/` characters and/or trailing `/tlpkg` and/or `/archive` components are ignored.

--gui [*action*]
Two notable GUI front-ends for `tlmgr`, `tlshell` and `tlcockpit`, are started up as separate programs; see their own documentation.

`tlmgr` itself has a graphical interface as well as the command line interface. You can give the option to invoke it, `--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. If no action is given, the GUI will be started at the main screen. See "GUI FOR TLMGR".

However, the native GUI requires Perl/TK, which is no longer included in TeX Live's Perl distribution for Windows. You may find `tlshell` or `tlcockpit` easier to work with.

--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), Ukrainian (uk), Vietnamese (vi), simplified Chinese (zh_CN), and traditional Chinese (zh_TW).

tlshell shares its message catalog with tlmgr.

--command-logfile *file*
`tlmgr` logs the output of all programs invoked (mktexlr, mtxrun, fmtutil, updmap) to a separate log file, by default `TEXMFSYSVAR/web2c/tlmgr-commands.log`. This option allows you to specify a different file for the log.

--debug-translation
In GUI mode, this switch tells `tlmgr` to report any untranslated (or missing) messages to standard error. This can help translators to see what remains to be done.

--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.

--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.

--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 specify 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 `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`.

--pin-file
Change the pinning file location from `TEXMFLOCAL/tlpkg/pinning.txt` (see "Pinning" below). Documented only for completeness, as this is only useful in debugging.

--usermode
Activates user mode for this run of `tlmgr`; see "USER MODE" below.

--usertree *dir*
Uses *dir* for the tree in user mode; see "USER MODE" below.

--verify-repo=[none|main|all]
Defines the level of verification done: If `none` is specified, no verification whatsoever is done. If `main` is given and a working GnuPG (`gpg`) binary is available, all repositories are checked, but only the main repository is required to be signed. If `all` is given, then all repositories need to be signed. See "CRYPTOGRAPHIC VERIFICATION" below for details.

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 also given, revision number for the loaded TeX Live Perl modules are shown, too.

ACTIONS

help
Display this help information and exit (same as `--help`, and on the web at <https://tug.org/texlive/doc/tlmgr.html>). Sometimes the `perldoc` and/or `PAGER` programs on the system have problems, resulting in control characters being literally output. This can't always be detected, but you can set the `NOPERLDOC` environment variable and `perldoc` will not be used.

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

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

backup
backup [*option*...] --all

backup [*option*...] *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. (The installer sets `backupdir` to `.../tlpkg/backups`, under the TL root installation directory, so it is usually defined; see the "option" description for more information.)

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.

candidates *pkg*

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

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

Execute one (or all) check(s) of the consistency of the installation. If no problems are found, there will be no output. (To get a view of what is being done, run `tlmgr -v check`.)

depends

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

files
Checks that all files listed in the local TLPDB (`texlive.tlpdb`) are actually present, and lists those missing.

runfiles
List those filenames that are occurring more than one time in the runfiles sections, except for known duplicates.

texmfdbs
Checks related to the `ls-R` files. If you have defined new trees, or changed the `TEXMF` or `TEXMFDBS` variables, it can't hurt to run this. It checks that:

- all items in `TEXMFDBS` have the `!!` prefix.
- all items in `TEXMFBDS` have an `ls-R` file (if they exist at all).
- all items in `TEXMF` with `!!` are listed in `TEXMFDBS`.
- all items in `TEXMF` with an `ls-R` file are listed in `TEXMFDBS`.

Options:

--use-svn
Use the output of `svn status` instead of listing the files; for checking the TL development repository. (This is run nightly.)

conf

conf [texmf|tlmgr|updmap [--conffile *file*] [--delete] [*key* [*value*]]]
conf auxtrees [--conffile *file*] [show|add|remove] [*value*]
With only `conf`, show general configuration information for TeX Live, including active configuration files, path settings, and more. This is like running `texconfig conf`, but works on all supported platforms.

With one of `conf texmf`, `conf tlmgr`, or `conf updmap`, shows all key/value pairs (i.e., all settings) as saved in `ROOT/texmf.cnf`, the user-specific `tlmgr` configuration file (see below), or the first found (via `kpsewhich`) `updmap.cfg` file, respectively.

If *key* is given in addition, shows the value of only that *key* in the respective file. If option *--delete* is also given, the value in the given configuration file is entirely removed (not just commented out).

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

The `PATH` value shown by `conf` is as used by `tlmgr`. The directory in which the `tlmgr` executable is found is automatically prepended to the PATH value inherited from the environment.

Here is a practical example of changing configuration values. 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
```

The subcommand `auxtrees` allows adding and removing arbitrary additional texmf trees, completely under user control. `auxtrees show` shows the list of additional trees, `auxtrees add` *tree* adds a tree to the list, and `auxtrees remove` *tree* removes a tree from the list (if present). The trees should not contain an `ls-R` file (or files will not be

found if the `ls-R` becomes stale). This works by manipulating the Kpathsea variable `TEXMFAUXTREES`, in (by default) `ROOT/texmf.cnf`. Example:

```
tlmgr conf auxtrees add /quick/test/tree
tlmgr conf auxtrees remove /quick/test/tree
```

In all cases the configuration file can be explicitly specified via the option `--conffile` *file*, e.g., if you don't want to change the system-wide configuration.

Warning: The general facility for changing configuration values is here, but tinkering with settings in this way is strongly discouraged. Again, no error checking on either keys or values is done, so any sort of breakage is possible.

dump-tlpdb [*option*...] [--json]

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.

--json
Instead of dumping the actual content, the database is dumped as JSON. For the format of JSON output see `tlpkg/doc/JSON-formats.txt`, format definition `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.

generate

generate [*option*...] language
generate [*option*...] language.dat
generate [*option*...] language.def
generate [*option*...] language.dat.lua

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 <https://tug.org/fonts/fontinstall.html>.

For managing your own formats, please read the `fmtutil --help` information.

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

The locally-maintained files are `language-local.dat`, `language-local.def`, or `language-local.dat.lua`, 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` and `fmtutil-local.cnf` are no longer read, since `updmap` and `fmtutil` now reads and supports multiple configuration files. Thus, local additions can and should be put into an `updmap.cfg` of `fmtutil.cnf` file in `TEXMFLOCAL`. The `generate updmap` and `generate fmtutil` actions no longer exist.)

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

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

where `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:

```
%!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)
```

gui

Start the graphical user interface. See **GUI** below.

info

info [*option*...] *pkg*...
info [*option*...] collections
info [*option*...] schemes

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, sizes, installation status, and TeX Live revision number. If *pkg* is not locally installed, searches in the remote installation source.

For normal packages (not collections or schemes), the sizes of the four groups of files (run/src/doc/bin files) are shown separately. For collections, the cumulative size is shown, including all directly-dependent packages (but not dependent collections). For schemes, the cumulative size is also shown, including all directly-dependent collections and packages.

If *pkg* is not found locally or remotely, the search action is used and lists matching packages and files.

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

--only-remote
Only list packages from the remote repository. Useful when checking what is available in a remote repository using `tlmgr --repo ... --only-remote info`. Note that `--only-installed` and `--only-remote` cannot both be specified.

--data item1,item2,...
If the option `--data` is given, its argument must be a comma separated list of field names from: `name`, `category`, `localrev`, `remoterev`, `shortdesc`, `longdesc`, `installed`, `size`, `relocatable`, `depends`, `cat-version`, `cat-date`, `cat-license`, plus various `cat-contact-*` fields (see below).

The `cat-*` fields all come from the TeX Catalogue (<https://ctan.org/pkg/catalogue>). For each, there are two more variants with prefix `l` and `r`, e.g., `lcat-version` and `rcat-version`, which indicate the local and remote information, respectively. The variants without `l` and `r` show the most current one, which is normally the remote value.

The requested packages' information is listed in CSV format, one package per line, and the column information is given by the `itemN`. The `depends` column contains the names of all the dependencies separated by `:` characters.

At this writing, the `cat-contact-*` fields include: `home`, `repository`, `support`, `bugs`, `announce`, `development`. Each may be empty or a url value. A brief description is on the CTAN upload page for new packages: <https://ctan.org/upload>.

--json
In case `--json` is specified, the output is a JSON encoded array where each array element is the JSON representation of a single `TLPOBJ` but with additional information. For details see `tlpkg/doc/JSON-formats.txt`, format definition: `TLPOBJINFO`. If both `--json` and `--data` are given, `--json` takes precedence.

init-usertree
Sets up a texmf tree for so-called user mode management, either the default user tree (`TEXMFHOME`), or one specified on the command line with `--usertree`. See "USER MODE" below.

install [*option***...]** ***pkg*...**
Install each *pkg* given on the command line, if it is not already installed. It does not touch existing packages; see the `update` action for how to get the latest version of a package.

By default this also installs all packages on which the given *pkg*s are dependent. Options:

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

--file
Instead of fetching a package from the installation repository, use the package files given on the command line. These files must be standard TeX Live package files (with contained tlpobj file).

--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.

--no-depends
Do not install dependencies. (By default, installing a package ensures that all dependencies of this package are fulfilled.)

--no-depends-at-all
Normally, 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 option suppresses this behavior, and also implies `--no-depends`. Don't use it unless you are sure of what you are doing.

--reinstall
Reinstall a package (including dependencies for collections) even if it already seems to be 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 (i.e., not those of category Scheme or Collection).

--with-doc
--with-src
While not recommended, the `install-tl` program provides an option to omit installation of all documentation and/or source files. (By default, everything is installed.) After such an installation, you may find that you want the documentation or source files for a given package after all. You can get them by using these options in conjunction with `--reinstall`, as in (using the `fontspec` package as the example):

```
tlmgr install --reinstall --with-doc --with-src fontspec
```

This action does not automatically add new symlinks in system directories; you need to run `tlmgr path add` ("path") yourself if you are using this feature and want new symlinks added.

key
key list
key add *file*
key remove *keyid*
The action `key` allows listing, adding and removing additional GPG keys to the set of trusted keys, that is, those that are used to verify the TeX Live databases.

With the `list` argument, `key` lists all keys.

The `add` argument requires another argument, either a filename or - for stdin, from which the key is added. The key is added to the local keyring `GNUPGHOME/repository-keys.gpg`, which is normally `tlpkg/gpg/repository-keys.gpg`.

The `remove` argument requires a key id and removes the requested id from the local keyring.

list
Synonym for "info".

option
option [--json] [show]
option [--json] showall|help
option *key* [*value*]

The first form, `show`, 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, `showall`, is similar, but also shows options which can be defined but are not currently set to any value (`help` is a synonym).

Both `show...` forms take an option `--json`, which dumps the option information in JSON format. In this case, both forms dump the same data. For the format of the JSON output see `tlpkg/doc/JSON-formats.txt`, format definition `TLOPTION`.

In the third form, with *key*, 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    (generate formats at installation or update 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 https://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 a synonym 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 how and want to regenerate formats yourself whenever needed (which is often, in practice).

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 to developers doing debugging.

The `docfiles` and `srcfiles` options control the installation of their respective file groups (documentation, sources; grouping is approximate) per package. By default both are enabled (1). Either or both can be disabled (set to 0) if disk space is limited or for minimal testing installations, etc. When disabled, the respective files are not downloaded at all.

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 TL installer sets `backupdir` to `.../tlpkg/backups`, under the TL root installation directory.

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. The default value is 1, so that backups are made, but only one backup is kept.

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

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

The `--` avoids having the `-1` treated as an option. (The `--` 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 systems to control the generation of links for executables, Info files and man pages. See the `path` action for details.

The last three options affect behavior 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.

paper

paper [a4|letter]
<[xdvi|pdftex|dvips|dvipdfmx|context|psutils] paper [*papersize*|--list]>
paper --json

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.

If `--json` is specified without other options, the paper setup is dumped in JSON format. For the format of JSON output see `tlpkg/doc/JSON-formats.txt`, format definition `TLPAPER`.

Incidentally, this syntax of having a specific program name before the `paper` keyword is unusual. 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.

path

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

On Unix, adds or removes symlinks for executables, 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. Furthermore, any

executables added or removed by future updates are not taken care of automatically; this command must be rerun as needed.

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.

pinning

The `pinning` action manages the pinning file, see "Pinning" below.

`pinning show`
Shows the current pinning data.

`pinning add` *repo pkgglob...*
Pins the packages matching the *pkgglob*(s) to the repository *repo*.

`pinning remove` *repo pkgglob...*
Any packages recorded in the pinning file matching the <pkgglob>s for the given repository *repo* are removed.

`pinning remove` *repo* `--all`
Remove all pinning data for repository *repo*.

platform

platform list|add|remove *platform*...
platform set *platform*
platform set auto

`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.

`platform set` *platform* switches TeX Live to always use the given platform instead of auto detection.

`platform set auto` switches TeX Live to auto detection mode for platform.

Platform detection is needed to select the proper `xz` and `wget` binaries that are shipped with TeX Live.

`arch` is a synonym for `platform`.

Options:

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

postaction

postaction [*option*...] install [shortcut|fileassoc|script] [*pkg*...]
postaction [*option*...] 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.

Options:

−−w32mode=[user|admin]
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=[1|2]
`−−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`.)

−−all
Carry out the postactions for all packages

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.

print-platform-info
Print the TeX Live platform identifier, TL platform long name, and original output from guess.

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:

−−all
Uninstalls all of TeX Live, asking for confirmation unless `−−force` is also specified.

−−backup
−−backupdir *directory*
These options behave just as with the update action (q.v.), except they apply to making backups of packages before they are removed. The default is to make such a backup, that is, to save a copy of packages before removal.

The ‘‘restore’’ action explains how to restore from a backup.

−−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.

Except with `--all`, this `remove` action does not automatically remove symlinks to executables from system directories; you need to run `tlmgr path remove` (“path”) yourself if you remove an individual package with a symlink in a system directory.

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

The first form, `repository list`, lists all configured repositories and the respective tags if set. If a path, url, or tag is given after the `list` keyword, it is interpreted as the source from which to initialize a TL database and lists the contained packages. This can also be an otherwise-unused repository, either local or remote. If the option `--with-platforms` is specified in addition, for each package the available platforms (if any) are also listed.

The form `repository add` adds a repository (optionally attaching a tag) to the list of repositories, while `repository remove` removes a repository, either by full path/url, or by tag.

The form `repository set` sets the list of available repositories to the items given on the command line, overwriting previous settings.

The form `repository status` reports the verification status of the loaded repositories with the format of one repository per line with fields separated by a single space:

The tag (which can be the same as the url);
= the url;

= iff machine-readable output is specified, the verification code (a number);

= a textual description of the verification status, as the last field extending to the end of

line.

That is, in normal (not machine-readable) output, the third field (numeric verification status) is not present.

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

restore

restore [*option*...] *pkg* [*rev*]
restore [*option*...] −−all

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).

If (and only if) both *pkg* and a valid revision number *rev* are specified, try 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.

−−json
When listing backups, the option `--json` turn on JSON output. The format is an array of JSON objects (`name`, `rev`, `date`). For details see `tlpkg/doc/JSON-formats.txt`, format definition: `TLBACKUPS`. If both `--json` and `--data` are given, `--json` takes precedence.

search

search [*option*...] *what*
search [*option*...] −−file *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 (Perl) regular expression.

Options:

−−file
List all filenames containing *what*.

−−all
Search everything: package names, descriptions and filenames.

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

−−word
Restrict the search of package names and descriptions (but not filenames) 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).

shell
Starts an interactive mode, where tlmgr prompts for commands. This can be used directly, or for scripting. The first line of output is `protocol` *n*, where *n* is an unsigned number identifying the protocol version (currently 1).

In general, tlmgr actions that can be given on the command line translate to commands in this shell mode. For example, you can say `update --list` to see what would be updated. The TLPDB is loaded the first time it is needed (not at the beginning), and used for the rest of the session.

Besides these actions, a few commands are specific to shell mode:

protocol
Print `protocol` *n*, the current protocol version.

help
Print pointers to this documentation.

version
Print tlmgr version information.

quit, end, bye, byebye, EOF
Exit.

restart
Restart `tlmgr shell` with the original command line; most useful when developing `tlmgr`.

load [local|remote]
Explicitly load the local or remote, respectively, TLPDB.

save
Save the local TLPDB, presumably after other operations have changed it.

get [*var*] =item set [*var* [*val*]]
Get the value of *var*, or set it to *val*. Possible *var* names: `debug-translation`, `machine-readable`, `no-execute-actions`, `require-verification`, `verify-downloads`, `repository`, and `prompt`. All except `repository` and `prompt` are booleans, taking values 0 and 1, and behave like the corresponding command line option. The `repository` variable takes a string, and sets the remote repository location. The `prompt` variable takes a string, and sets the current default prompt.

If *var* or then *val* is not specified, it is prompted for.

show
Synonym for ‘‘info’’.

uninstall
Synonym for remove.

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. If updates to `tlmgr` itself are present, this gives 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` (not recommended, see option description).

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` (also not recommended).

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`.

To reiterate: automatic removals and additions are entirely determined by comparison of collections. Thus, if you manually install an individual package `foo` which is later removed from the server, `tlmgr` will not notice and will not remove it locally. (It has to be this way, without major rearchitecture work, because the tlpdb does not record the repository from which packages come from.)

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.

This option can also be set permanently in the tlmgr config file with the key `update-exclude`.

--no-auto-remove [*pkg*...]
By default, `tlmgr` tries to remove packages in an existing collection which have disappeared on the server, as described above under `--all`. This option prevents such removals, either for all packages (with `--all`), or for just the given *pkg* names. This can lead to an inconsistent TeX installation, since packages are not infrequently renamed or replaced by their authors. Therefore this is not recommended.

--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
```

Again, since packages are sometimes renamed or replaced, using this option is not recommended.

--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

--backupdir *directory*

These two options control the creation of backups of packages *before* updating; that is, backing up packages as currently installed. If neither option is given, no backup will made. 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 also set options via the "option" action to automatically make backups for all packages, and/or keep only a certain number of backups.

`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 save a persistent backup in case the actual *content* of the update causes problems, e.g., introduces an TeX 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.

`tlmgr` saves one copy of the main `texlive.tlpdb` file used for an update with a suffix representing the repository url, as in `tlpkg/texlive.tlpdb.main.`*long-hash-string*. Thus, even when many mirrors are used, only one main `tlpdb` backup is kept. For non-main

repositories, which do not generally have (m)any mirrors, no pruning of backups is done.

This action does not automatically add or remove new symlinks in system directories; you need to run `tlmgr` "path" yourself if you are using this feature and want new symlinks added.

CONFIGURATION FILE FOR TLMGR

`tlmgr` reads two configuration files: one is system-wide, in `TEXMFSYSCONFIG/tlmgr/config`, and the other is user-specific, in `TEXMFCONFIG/tlmgr/config`. The user-specific one is the default for the `conf tlmgr` action. (Run `kpsewhich -var-value=TEXMFSYSCONFIG` or `... TEXMFCONFIG ...` to see the actual directory names.)

A few defaults corresponding to command-line options can be set in these configuration files. In addition, the system-wide file can contain a directive to restrict the allowed actions.

In these config files, empty lines and lines starting with # are ignored. All other lines must look like:

```
key = value
```

where the spaces are optional but the = is required.

The allowed keys are:

`auto-remove`, value 0 or 1 (default 1), same as command-line option.
`gui-expertmode`, value 0 or 1 (default 1). This switches between the full GUI and a simplified GUI with only the most common settings.
`gui-lang` *llcode*, with a language code value as with the command-line option.
`no-checksums`, value 0 or 1 (default 0, see below).
`persistent-downloads`, value 0 or 1 (default 1), same as command-line option.
`require-verification`, value 0 or 1 (default 0), same as command-line option.
`tkfontscale`, value any float. Controls the scaling of fonts in the Tk based frontends.
`update-exclude`, value: comma-separated list of packages (no space allowed). Same as the command line option `--exclude` for the action `update`.
`verify-downloads`, value 0 or 1 (default 1), same as command-line option.

The system-wide config file can contain one additional key:

`allowed-actions` *action1* [,*action*,...] The value is a comma-separated list of `tlmgr` actions which are allowed to be executed when `tlmgr` is invoked in system mode (that is, without `--usermode`).

This allows distributors to include the `tlmgr` in their packaging, but allow only a restricted set of actions that do not interfere with their distro package manager. For native TeX Live installations, it doesn't make sense to set this.

The `no-checksums` key needs more explanation. By default, package checksums computed and stored on the server (in the TLPDB) are compared to checksums computed locally after downloading. `no-checksums` disables this process.

The checksum algorithm is SHA-512. Your system must have one of (looked for in this order) the Perl `Digest::SHA` module, the `openssl` program (<https://openssl.org>), the `sha512sum` program (from GNU Coreutils, <https://www.gnu.org/software/coreutils>), or finally the `shasum` program (just to support old Macs). If none of these are available, a warning is issued and `tlmgr` proceeds without checking checksums. (Incidentally, other SHA implementations, such as the pure Perl and pure Lua modules, are much too slow to be usable in

our context.) `no-checksums` avoids the warning.

CRYPTOGRAPHIC VERIFICATION

`tlmgr` and `install-tl` perform cryptographic verification if possible. If verification is performed and successful, the programs report `(verified)` after loading the TLPDB; otherwise, they report `(not verified)`. But either way, by default the installation and/or updates proceed normally.

If a program named `gpg` is available (that is, found in `PATH`), by default cryptographic signatures will be checked: we require the main repository be signed, but not any additional repositories. If `gpg` is not available, by default signatures are not checked and no verification is carried out, but `tlmgr` still proceeds normally.

The behavior of the verification can be controlled by the command line and config file option `verify-repo` which takes one of the following values: `none`, `main`, or `all`. With `none`, no verification whatsoever is attempted. With `main` (the default) verification is required only for the main repository, and only if `gpg` is available; though attempted for all, missing signatures of subsidiary repositories will not result in an error. Finally, in the case of `all`, `gpg` must be available and all repositories need to be signed.

In all cases, if a signature is checked and fails to verify, an error is raised.

Cryptographic verification requires checksum checking (described just above) to succeed, and a working GnuPG (`gpg`) program (see below for search method). Then, unless cryptographic verification has been disabled, a signature file (`texlive.tlpdb.*.asc`) of the checksum file is downloaded and the signature verified. The signature is created by the TeX Live Distribution GPG key 0x0D5E5D9106BAB6BC, which in turn is signed by Karl Berry's key 0x0716748A30D155AD and Norbert Preining's key 0x6CACA448860CDC13. All of these keys are obtainable from the standard key servers.

Additional trusted keys can be added using the `key` action.

Configuration of GnuPG invocation

The executable used for GnuPG is searched as follows: If the environment variable `TL_GNUPG` is set, it is tested and used; otherwise `gpg` is checked; finally `gpg2` is checked.

Further adaptation of the `gpg` invocation can be made using the two environment variables `TL_GNUPGHOME`, which is passed to `gpg` as the value for `--homedir`, and `TL_GNUPGARGS`, which replaces the default options `--no-secmem-warning --no-permission-warning`.

USER MODE

`tlmgr` provides a restricted way, called ''user mode'', to manage arbitrary texmf trees in the same way as the main installation. For example, this allows people without write permissions on the installation location to update/install packages into a tree of their own.

`tlmgr` is switched into user mode with the command line option `--usermode`. It does not switch automatically, nor is there any configuration file setting for it. Thus, this option has to be explicitly given every time user mode is to be activated.

This mode of `tlmgr` works on a user tree, by default the value of the `TEXMFHOME` variable. This can be overridden with the command line option `--usertree`. In the following when we speak of the user tree we mean either `TEXMFHOME` or the one given on the command line.

Not all actions are allowed in user mode; `tlmgr` will warn you and not carry out any

problematic actions. Currently not supported (and probably will never be) is the `platform` action. The `gui` action is currently not supported, but may be in a future release.

Some `tlmgr` actions don't need any write permissions and thus work the same in user mode and normal mode. Currently these are: `check`, `help`, `list`, `print-platform`, `print-platform-info`, `search`, `show`, `version`.

On the other hand, most of the actions dealing with package management do need write permissions, and thus behave differently in user mode, as described below: `install`, `update`, `remove`, `option`, `paper`, `generate`, `backup`, `restore`, `uninstall`, `symlinks`.

Before using `tlmgr` in user mode, you have to set up the user tree with the `init-usertree` action. This creates *usertree*`/web2c` and *usertree*`/tlpkg/tlpobj`, and a minimal *usertree*`/tlpkg/texlive.tlpdb`. At that point, you can tell `tlmgr` to do the (supported) actions by adding the `--usermode` command line option.

In user mode the file *usertree*`/tlpkg/texlive.tlpdb` contains only the packages that have been installed into the user tree using `tlmgr`, plus additional options from the ''virtual'' package `00texlive.installation` (similar to the main installation's `texlive.tlpdb`).

All actions on packages in user mode can only be carried out on packages that are known as `relocatable`. This excludes all packages containing executables and a few other core packages. Of the 2500 or so packages currently in TeX Live the vast majority are relocatable and can be installed into a user tree.

Description of changes of actions in user mode:

User mode install

In user mode, the `install` action checks that the package and all dependencies are all either relocated or already installed in the system installation. If this is the case, it unpacks all containers to be installed into the user tree (to repeat, that's either `TEXMFHOME` or the value of `--usertree`) and add the respective packages to the user tree's `texlive.tlpdb` (creating it if need be).

Currently installing a collection in user mode installs all dependent packages, but in contrast to normal mode, does *not* install dependent collections. For example, in normal mode `tlmgr install collection-context` would install `collection-basic` and other collections, while in user mode, *only* the packages mentioned in `collection-context` are installed.

If a package shipping map files is installed in user mode, a backup of the user's `updmap.cfg` in `USERTREE/web2c/` is made, and then this file regenerated from the list of installed packages.

User mode backup, restore, remove, update

In user mode, these actions check that all packages to be acted on are installed in the user tree before proceeding; otherwise, they behave just as in normal mode.

User mode generate, option, paper

In user mode, these actions operate only on the user tree's configuration files and/or `texlive.tlpdb`. creates configuration files in user tree

MULTIPLE REPOSITORIES

The main TeX Live repository contains 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, for whatever reason.

The simplest and most reliable method is to temporarily set the installation source to any repository (with the `-repository` or `option repository` command line options), and perform your operations.

When you are using multiple repositories over a sustained length of time, however, explicitly switching between them becomes 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. When you switch from a single repository installation to a multiple repository installation, the previous sole repository will be set as the main repository.

By default, even if multiple repositories are configured, packages are *still 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 the new repository, by specifying so-called ''pinning'' rules, described next.

Pinning

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.

As mentioned above, by default everything is pinned to the main repository. Let's now go through an example of setting up a second repository and enabling updates of a package from it.

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
```

Ok. Let's add the `tlcontrib` repository (this is a real repository hosted at <http://contrib.texlive.info>) with the tag `tlcontrib`:

```
$ tlmgr repository add http://contrib.texlive.info/current tlcontrib
```

Check the repository list again:

```
$ tlmgr repository list
List of repositories (with tags if set):
   http://contrib.texlive.info/current (tlcontrib)
   /var/www/norbert/tlnet (main)
```

Now we specify a pinning entry to get the package `classico` from `tlcontrib`:

```
$ tlmgr pinning add tlcontrib classico
```

Check that we can find `classico`:

```
$ tlmgr show classico
package:     classico
...
shortdesc:   URW Classico fonts
...
```

– install `classico`:

```
$ tlmgr install classico
tlmgr: package repositories:
...
[1/1,  ??:??/??:??] install: classico @tlcontrib [737k]
```

In the output here you can see that the `classico` package has been installed from the `tlcontrib` repository (`@tlcontrib`).

Finally, `tlmgr pinning` also supports removing certain or all packages from a given repository:

```
$ tlmgr pinning remove tlcontrib classico # remove just classico
$ tlmgr pinning remove tlcontrib --all    # take nothing from tlcontrib
```

A summary of `tlmgr pinning` actions is given above.

GUI FOR TLMGR

The graphical user interface for `tlmgr` requires Perl/Tk <https://search.cpan.org/search?query=perl%2Ftk>. For Unix-based systems Perl/Tk (as well as Perl of course) has to be installed outside of TL. <https://tug.org/texlive/distro.html#perltk> has a list of invocations for some distros. For Windows the necessary modules are no longer shipped within TeX Live, so you'll have to have an external Perl available that includes them.

We are talking here about the GUI built into tlmgr itself, not about the other tlmgr GUIs, which are: tlshell (Tcl/Tk−based), tlcockpit (Java-based) and, only on Macs, TeX Live Utility. These are invoked as separate programs.

The GUI mode of tlmgr is started with the invocation `tlmgr gui`; assuming Tk is loadable, 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 searches both descriptions and filenames. You can also select a 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 TLMGR" 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 <https://tug.org/texlive/doc.html>) and the usual "About" box.

GUI options

Some generic Perl/Tk options can be specified with `tlmgr gui` to control the display:

`-background` *color*
: Set background color.

`-font` ``*fontname fontsize*''
: Set font, e.g., `tlmgr gui -font "helvetica 18"`. The argument to `-font` must be quoted, i.e., passed as a single string.

`-foreground` *color*
: Set foreground color.

`-geometry` *geomspec*
: Set the X geometry, e.g., `tlmgr gui -geometry 1024x512-0+0` creates the window of (approximately) the given size in the upper-right corner of the display.

`-xrm` *xresource*
: Pass the arbitrary X resource string *xresource*.

A few other obscure options are recognized but not mentioned here. See the Perl/Tk documentation (<https://search.cpan.org/perldoc?Tk>) for the complete list, and any X documentation for general information.

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, install, and ''option'' actions.

Machine-readable `update` and `install` output

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.

Machine-readable `option` output

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.

ENVIRONMENT VARIABLES

`tlmgr` uses many of the standard TeX environment variables, as reported by, e.g., `tlmgr conf` ("conf").

In addition, for ease in scripting and debugging, `tlmgr` looks for the following environment variables. These are not of interest for normal user installations.

`TEXLIVE_COMPRESSOR`

This variable allows selecting a different compressor program for backups and intermediate rollback containers. The order of selection is:

1. If the environment variable `TEXLIVE_COMPRESSOR` is defined, use it; abort if it doesn't work. Possible values: `lz4`, `gzip`, `xz`. The necessary options are added internally.
2. If lz4 is available (either from the system or TL) and working, use that.
3. If gzip is available (from the system) and working, use that.
4. If xz is available (either from the system or TL) and working, use that.

lz4 and gzip are faster in creating tlmgr's local backups, hence they are preferred. The unconditional use of xz for the tlnet containers is unaffected, to minimize download sizes.

`TEXLIVE_DOWNLOADER`
`TL_DOWNLOAD_PROGRAM`
`TL_DOWNLOAD_ARGS`

These options allow selecting different download programs then the ones automatically selected by the installer. The order of selection is:

1. If the environment variable `TEXLIVE_DOWNLOADER` is defined, use it; abort if the specified program doesn't work. Possible values: `lwp`, `curl`, `wget`. The necessary options are added internally.
2. If the environment variable `TL_DOWNLOAD_PROGRAM` is defined (can be any value), use it together with `TL_DOWNLOAD_ARGS`; abort if it doesn't work.
3. If LWP is available and working, use that (by far the most efficient method, as it supports persistent downloads).
4. If curl is available (from the system) and working, use that.
5. If wget is available (either from the system or TL) and working, use that.

TL provides `wget` binaries for platforms where necessary, so some download method should always be available.

`TEXLIVE_PREFER_OWN`

By default, compression and download programs provided by the system, i.e., found along `PATH` are preferred over those shipped with TeX Live.

This can create problems with systems that are too old, and so can be overridden by setting the environment variable `TEXLIVE_PREFER_OWN` to 1. In this case, executables shipped with TL will be preferred.

Extra compression/download programs not provided by TL, such as gzip, lwp, and curl, are still checked for on the system and used if available, per the above.
`TEXLIVE_PREFER_OWN` only applies when the program being checked for is shipped

with TL, namely the lz4 and xz compressors and wget downloader.

Exception: on Windows, the `tar.exe` shipped with TL is always used, regardless of any setting.

AUTHORS AND COPYRIGHT

`$Id:` tlmgr.pl 59064 2021−05−03 17:37:44Z karl $