LaTeX Plugin for Sublime Text 2 and 3

by Ian Bacher, Marciano Siniscalchi, and Richard Stein

Marciano's blog:
http://tekonomist.wordpress.com

Additional contributors (*thank you thank you thank you*): first of all, Wallace Wu and Juerg Rast, who contributed code for multifile support in ref and cite completions, "new-style" ref/cite completion, and project file support. Also, skuroda (Preferences menu), Sam Finn (initial multifile support for the build command); Daniel Fleischhacker (Linux build fixes), Mads Mobaek (universal newline support), Stefan Ollinger (initial Linux support), RoyalTS (aka Tobias Schidt?) (help with bibtex regexes and citation code, various fixes), Juan Falgueras (latexmk option to handle non-ASCII paths), Jeremy Jay (basic biblatex support), Ray Fang (texttt snippet), Ulrich Gabor (tex engine selection and cleaning aux files), Wes Campaigne and 'jlegewie' (ref/cite completion 2.0!). **Huge** thanks to Daniel Shannon (aka phyllisstein) who first ported LaTeXTools to ST3. Also thanks for Charley Peng, who has been assisting users and generating great pull requests; I'll merge them as soon as possible. Also William Ledoux (various Windows fixes, env support), Sean Zhu (find Skim.app in non-standard locations), Maximilian Berger (new center/table snippet), Lucas Nanni (recursively delete temp files), Sergey Slipchenko (`$` auto-pairing with Vintage), btstream (original fill-all command; LaTeX-cwl support), Richard Stein (auto-hide build panel, jump to included tex files, LaTeX-cwl support config, TEX spellcheck support, functions to analyze LaTeX documents, cache functionality, multiple cursor editing), Dan Schrage (nobibliography command), PoByBolek (more biblatex command), Rafael Lerm (support for multiple lines in `\bibliography` commands), Jeff Spencer (override keep_focus and forward_sync via key-binding), Jonas Malaco Filho (improvements to the Evince scripts), Michael Bar-Sinai (bibtex snippets).

If you have contributed and I haven't acknowledged you, email me!

Latest revision: v3.10.2 (2016-08-04).

Headline features:

- Better curly-brace completion algorithm
- Support for more ref commands
- CWL files can now be used based on loaded packages
- Support for input completions following symlinks

Contents

1 Introduction

This plugin provides several features that simplify working with LaTeX files:

- The ST build command takes care of compiling your LaTeX source to PDF using `texify` (Windows/MikTeX) or `latexmk` (OSX/MacTeX, Windows/-TeXlive, Linux/TeXlive). Then, it parses the log file and lists errors and warning. Finally, it launches (or refreshes) the PDF viewer (SumatraPDF on Windows, Skim on OSX, and Evince on Linux by default) and jumps to the current cursor position.
- Forward and inverse search with the named PDF previewers is fully supported
- Easy insertion of references and citations (from BibTeX files)
- Plugs into the "Goto anything" facility to make jumping to any section or label in your LaTeX file(s)
- Smart command completion for a variety of text and math commands is provided
- Additional snippets and commands are also provided
- The build command is fully customizable, as is the PDF previewer.

2 Bugs, issues & feature requests

Please read the Requirements and Setup section carefully to ensure you get up and running as quickly as possible. Help for troubleshooting common issues can be found in the Troubleshooting section at the end of this README. For other bugs, issues or to request new features, please get in touch with us via Github.

Please search for existing issues and pull requests before opening a new issue.

3 Requirements and Setup

First, you need to be running Sublime Text 2 or 3 (ST2 and ST3 henceforth, or simply ST to refer to either ST2 or ST3). For ST3, this has been tested against the latest beta builds.

Second, get the LaTeXTools plugin. These days, the easiest way to do so is via Package Control. See here for details on how to set it up (it's very easy). Once you have Package Control up and running, invoke it (via the Command Palette from the Tools menu, or from Preferences), select the Install Package command, and look for LaTeXTools.

If you prefer a more hands-on approach, you can always clone the git repository, or else just grab this plugin's .zip file from GitHub and extract it to your Packages directory (you can open it easily from ST, by clicking on **Preferences|Browse Packages**). Then, (re)launch ST. Please note that if you do a manual installation, the Package **must** be named "LaTeXTools".

I encourage you to install Package Control anyway, because it's awesome, and it makes it easy to keep your installed packages up-to-date (see the previously linked page for details).

Third, follow the OS-specific instructions below.

Finally, look at the section on Platform-Specific Settings and ensure that all your settings are correct. If you are running LaTeXTools for the first time, you may want to run the `LaTeXTools: Reset user settings to default` command from the Command Palette to get an editable copy of the settings file.

3.1 OS X

On **OSX**, you need to be running the MacTeX distribution (which is pretty much the only one available on the Mac anyway). Just download and install it in the usual way. I have tested MacTeX versions 2010–2014, both 32 and 64 bits; these work fine. MacTeX 2015 also works. On the other hand, MacTeX 2008 does *not* seem to work out of the box (compilation fails), so please upgrade.

We recommend that you also install the Skim PDF viewer, as this provides forward and inverse search and is the default viewer that LaTeXTools uses on OS X. If you don't install Skim, please see the section on Viewers below for details on how to setup a viewer.

3.1.1 Setup Skim.app

To configure inverse search, open the Preferences dialog of the Skim.app, select the Sync tab, then:

- uncheck the "Check for file changes" option
- choose the Sublime Text 2 or Sublime Text 3 preset (yes, Skim now supports both ST2 and ST3 by default!)

In case you are using an old version of Skim, you can always choose the Custom preset and enter (for ST3):

```
/Applications/Sublime Text.app/Contents/SharedSupport/bin/subl
```

in the Command field, and `"%file":%line` in the Arguments field. (This is correct as of 7/18/2013; you may want to double-check that ST3 is indeed in `/Applications/Sublime Text.app`; just go to the Applications folder in the Finder. Adapt as needed for ST2).

3.1.2 Setup LaTeXTools

Finally, edit the file `LaTeXTools.sublime-settings` in the `User` directory to make sure that the configuration reflects your preferred TeX distribution. Open the file and scroll down to the section titled "Platform settings." Look at the block for your OS, namely `"osx"`. Within that block, verify that the `"texpath"` setting is correct. Note that `"texpath"` **must** include `$PATH` somewhere.

3.1.3 Support for BasicTeX

If you don't want to install the entire MacTeX distro, which is pretty big, BasicTeX will also work (of course, as long as the LaTeX packages you need

are included). **However**, you need to explicitly add the `latexmk` utility, which is not included by default: from the Terminal, type `sudo tlmgr install latexmk` (you will need to provide your password, assuming you are Administrator on your machine).

3.1.4 El Capitan

Sadly, with each OS X release, Apple deviates more and more from established Unix conventions. The latest "innovation" is that, beginning with El Capitan, applications can no longer write to `/usr`. MacTeX 2015 remedies this by creating a link to TeX binaries in `/Library/TeX`. The default LaTeXTools settings file now adds `/Library/TeX/texbin` to the `texpath`. In practice, this means the following.

3.2 Windows

On **Windows**, both MikTeX and TeXLive are supported. Install either of these as usual.

We recommend that you install a version of Sumatra PDF viewer, as this is the only viewer currently supported on Windows. Its very light-weight and supports both forward and inverse search. Just download and install it in the normal way. You may have to add the SumatraPDF directory to your `PATH` environment variable or else set the `sumatra` command in the `windows` platform settings (see the section on platform settings below). If you choose not to install SumatraPDF, you might be able to use the `command` viewer to support another PDF viewer. See the Viewers section below for details.

3.2.1 Setup Sumatra

You now need to set up inverse search in Sumatra PDF. However, the GUI for doing this is hidden in Sumatra until you open a PDF file that has actual synchronization information (that is, an associated `.synctex.gz` file): see here. If you have one such file, then open it, go to **Settings|Options**, and enter `"C:\Program Files \Sublime Text 2\sublime_text.exe" "%f:%l"` for ST2, and `"C:\Program Files\Sublime Text 3\sublime_text.exe" "%f:%l"` for ST3, as the inverse-search command line (in the text-entry field at the bottom of the options dialog). If you don't already have a file with sync information, you can easily create one: compile any LaTex file you already have (or create a new one) with `pdflatex -synctex=1 <file.tex>`, and then open the resulting PDF file in SumatraPDF.

As an alternative, you can open a command-line console (run `cmd.exe`), and issue the following command (this assumes that SumatraPDF.exe is in your path; replace 3 with 2 for ST2 of course):

```
sumatrapdf.exe -inverse-search "\"C:\Program Files\Sublime Text 3\
    sublime_text.exe\" \"%f:%l\""
```

I'm sorry this is not straightforward—it's not my fault :-)

3.2.2 Setup LaTeXTools

Finally, edit the file `LaTeXTools.sublime-settings` in the `User` directory to make sure that the configuration reflects your preferred TeX distribution. Open the file and scroll down to the section titled "Platform settings." Look at the block for your OS, namely `windows`. Within that block, verify that the `texpath` setting is correct; for MiKTeX, you can leave this empty, i.e., `""`. If you do specify a path, note that it **must** include the system path variable, i.e., `$PATH` (this syntax seems to be OK). Also verify that the `distro` setting is correct: the possible values are `"miktex"` and `"texlive"`.

TeXlive has one main advantage over MikTeX: it supports file names and paths with spaces.

3.2.3 `PATH` Issues

Recent versions of MikTeX add themselves to your path automatically, but in case the build system does not work, that's the first thing to check. TeXLive can also add itself to your path.

3.3 Linux

Linux support is coming along nicely. However, as a general rule, you will need to do some customization before things work. This is due to differences across distributions (a.k.a. "fragmentation"). Do not expect things to work out of the box.

3.3.1 Install TeXLive

You need to install TeXlive; if you are on Ubuntu, note that `apt-get install texlive` will get you a working but incomplete setup. In particular, it will *not* bring in `latexmk`, which is essential to LaTeXTools. You need to install it via `apt-get install latexmk`. If on the other hand you choose to install the TeXlive distro from TUG, `latexmk` comes with it, so you don't need to do anything else.

3.3.2 Setup LaTeXTools

You also need to edit the file `LaTeXTools.sublime-settings` in the `User` directory to make sure that the configuration reflects your preferred TeX distribution. Open that file and scroll down to the section titled "Platform settings." Look at the block for your OS, namely `"linux"`. Within that block, verify that the `"texpath"` setting is correct. Notice that this **must** include `$PATH` somewhere, or things will not work.

You may also have to set the `command` option in `"builder_settings"`, which tells the builder how to invoke `latexmk`. By default (i.e., if `command` is empty or not given) it is `["latexmk", "-cd", "-e", "-f", "-pdf", "-interaction=nonstopmode", "-synctex=1"]`.

If you customize the command to include a custom PDF command, users have reported the following possible issues and fixes (thanks!), so if you get a "Cannot compile!" error, try the following:

- some distros do *not* want a space before and after the `=` in `$pdflatex = %E`. But some *do* want the space there (sigh!)
- sometimes `latexmk` is not on the `PATH`, or the path is not correctly picked up by ST. In this case, instead of `"latexmk"`, use `"/usr/bin/latexmk"` or wherever `latexmk` is in your system.
- some distros require quoting the `$pdflatex` assignment, as in `"$pdflatex = \"'%E -interaction=nonstopmode -synctex=1 %S %O'\""`

There are several variants to deal with; each distro is a little bit different, so there are basically no universal defaults. There's not much I can do about it. Good luck!

Also, to get inverse search working on ST3, make sure you set the `sublime` option in `LaTeXTools.sublime-settings` correctly; the Ubuntu package from the ST web page uses `subl`, but check from the command line first.

3.3.3 Setup Evince

By default LaTeXTools assumes you are using Evince (Document Viewer) as your PDF viewer. Support is also available for Okular and other viewers that can be run via the command line. See the section on Viewers below for details on how to setup other viewers.

If you opt to use Evince, which is installed by default on Ubuntu and any distro that provides the Gnome desktop. Please note that you will also need to have the Python `dbus` bindings installed for your system version of Python. If you use a Gnome desktop, this is likely already installed, but if not, you will need to install it using your distro's package manager. In particular, it's been reported not to be installed by Arch Linux by default. Backward and forward search Work For Me (TM). Hopefully they will work for you, too, but let me know if this is not the case.

4 General Features

4.1 Project Files

Project files are fully supported! You should consult the subsection on project-specific settings for further details.

4.2 Multi-file documents

Multi-file documents are supported as follows. If the first line in the current file consists of the text `%!TEX root = <master file name>`, then tex & friends are invoked on the specified master file, instead of the current one. Note: the only file that gets saved automatically is the current one. Also, the master file

name **must** have a valid tex extension (i.e., one configured in the `tex_file_exts` settings), or it won't be recognized.

As an alternative, to using the `%!TEX root = <master file name>` syntax, if you use a Sublime project, you can set the `TEXroot` option (under `settings`):

```
{
    ... <folder-related settings> ...

    "settings": {
        "TEXroot": "yourfilename.tex"
    }
}
```

Note that if you specify a relative path as the `TEXroot` in the project file, the path is determined *relative to the location of the project file itself*. It may be less ambiguous to specify an absolute path to the `TEXroot` if possible.

4.3 Spell-checking

LaTeXTools parses the `%!TEX spellcheck` directive to set the language for the spell-checker integrated in Sublime Text. The Dictionaries package is recommended and supported. If you have additional dictionaries, you can add them using the `tex_spellcheck_paths` setting, which is a mapping from the locales to the dictionary paths. Each locale must be lowercase and use only a hyphen as a separator. The dictionary paths must be compatible with those used by Sublime Text's spell-checker. For example `{"en-us": "Packages/Language - English/en_US.dic"}` would be a valid value. For more on Sublime Text support for spell checking, see the relevant online documentation.

4.4 Support for non-`.tex` files

LaTeXTools has some degree of support for LaTeX documents that are in files with an extension other than `.tex`. In particular, this feature is designed to work well with alternative extensions, such as `.ltx`. Other extensions such as `.Rnw` and `.tikz` are supported, but, for now they will be treated as standard LaTeX documents (patches are always welcome!).

This behaviour is controlled through two settings, `tex_file_exts` and `latextools_set_syntax`. For more on these settings, see the documentation on General Settings.

Note that while the extension detection is used by features of LaTeXTools, including, other features—especially the completions—depend on the syntax of the file being set to `LaTeX` as well.

4.5 Support for Editing Bibliographies

LaTeXTools has some enhanced support for editing either BibTeX or BibLaTeX `.bib` files. Snippet completions are provided for every entry type supported by either BibTeX or BibLaTeX, as are completions for field names. In addition,

LaTeXTools provides smart completions for name fields (such as `author`, `editor`, etc.) and crossrefs. When auto-completions are triggered in a name field, a list of all entered names in the current file is presented. Similarly, when auto-completions are triggered in a crossref field, a list of all current entry keys will be provided.

This behaviour is controlled by a single setting, `use_biblatex` (default: `false`), which indicates whether LaTeXTools should use the BibTeX versions of the auto-completions (this is the default behavior) or the BibLaTeX versions of the auto-completions (if `use_biblatex` is set to `true`).

4.6 Package Documentation

You can consult the documentation for any LaTeX package by invoking the `View Package Documentation` command via the Command Palette (for now). This relies on your system's `texdoc` command.

4.7 Caching

LaTeXTools uses a cache to store relevant information about your document and improve the performance of commands. The contents of this cache may become outdated. If a command seems to be returning old data, simply clear the cache using either the command delete temporary files or the dedicated command to clear the cache. For more details on the cache, see the Cache settings section below and the section on the Cache.

5 Builder Features

Most of the builder features are controlled through the `LaTeXTools.sublime-settings` file. See, in particular, the section on builder settings.

5.1 Default Builder

The default builder (called the `traditional` builder) supports several additional features.

5.1.1 TeX Engine Selection

If the first line of the current file consists of the text `%!TEX program = <program>`, where `program` is `pdflatex`, `lualatex` or `xelatex`, the corresponding engine is selected. If no such directive is specified, `pdflatex` is the default. Multi-file documents are supported: the directive must be in the *root* (i.e. master) file. Also, for compatibility with TeXshop, you can use `TS-program` instead of `program`.

Note: for this to work, you must **not** customize the `command` option in `LaTeXTools.sublime-settings`. If you do, you will not get this functionality. Finally, if you use project files, the `program` builder setting can also be customized there, under `settings`.

5.1.2 TeX Options

You can pass command-line options to your engine in two ways (thanks Ian Bacher!). One is to use a `%!TEX options = ...` line at the top of your file. The other is to use the `options` builder setting in your settings file. This can be useful, for instance, if you need to allow shell escape. Finally, if you use project files, the `options` builder setting can also be customized there (again, under `settings`).

5.1.3 Output Directory and Auxiliary Directory

The `--output-directory` and `--aux-directory` flags can be set in several ways:

- Using a TEX directive, such as `%!TEX output_directory = <path>` near the top of the file.
- Using the TeX Options feature to set `--output-directory` and / or `--aux-directory`.
- Using the corresponding `output_directory` and `aux_directory` settings detailed in the settings section.

There are three special values that can be used, `<<temp>>` `<<project>>` and `<<cache>>`. Their meaning is the same as that found in the settings section and they are described there.

Note: the `--aux-directory` flag is only meaningful when used with MiKTeX, but see the `copy_on_build` setting for an option to get some similar behaviour with TeXLive.

Note: These flags can only be set when using latexmk (default on OS X and Linux), the `basic` builder or the `script` builder (see below for documentation on using the script builder). If you are using texify (default when using MiKTeX) or the simple builder, setting these flags through any method will be ignored.

5.1.4 Jobname

The `--jobname` flag can be set in several ways:

- Using a TEX directive, such as `%!TEX jobname = <jobname>` near the top of the file.
- Using the TeX Options feature to set `--jobname`
- Using the corresponding `jobname` setting detailed in the settings section.

Note: Jobname can only be set when using latexmk (default on OS X and Linux), the `basic` builder or the `script` builder (see below for documentation on using the script builder). If you are using texify (default when using MiKTeX) or the simple builder, setting the jobname flag through any method will be ignored.

5.1.5 Customizing the compilation command

It is possible to customize the command run by setting the `command` option under Builder Settings. See the section on Builder Settings for details.

Note: If you customize the command, the TeX engine selection facility may no longer work because it relies on a specific compilation command. However,

if you want to customize or replace `latexmk`/`texify`, you probably know how to select the right TeX engine, so this shouldn't be a concern. Also note that if you are using `latexmk` and you set the `$pdflatex` variable, the TeX options facility will not function, as `latexmk` does not support this.

If you change the compilation command, you are responsible for making it work on your setup. Only customize the compilation command if you know what you're doing.

5.2 Other Builders

If the default builder doesn't meet your needs for any reason, please see the section on Alternative Builders below.

6 Viewers

By default, LaTeXTools supports the following viewers, depending on platform:

- On OS X, Skim
- On Windows, Sumatra
- On Linux, Evince

However, it is possible to use other programs to view PDF files. Currently, there are viewers available for Preview.app, Okular and Zathura. These viewers can be chosen by changing the `"viewer"` setting. See the Viewer Settings section for details. If you are using an alternate viewer, please see the relevant section under Alternate Viewers for any caveats or other instructions. In addition, there is a viewer, called the Command Viewer which can be used to launch a PDF document using the command line.

7 Keybindings

Keybindings have been chosen to make them easier to remember, and also to minimize clashes with existing (and standard) ST bindings. I am taking advantage of the fact that ST supports key combinations, i.e. sequences of two (or more) keys. The basic principle is simple:

- **Most LaTeXTools facilities are triggered using `Ctrl+l` (Windows, Linux) or `Cmd+l` (OS X), followed by some other key or key combination**
- Compilation uses the standard ST "build" keybinding, i.e. `Ctrl-b` on Windows and Linux and `Cmd-b` on OS X. So does the "goto anything" facility (though this may change).

For example: to jump to the point in the PDF file corresponding to the current cursor position, use `Ctrl-l, j`: that is, hit `Ctrl-l`, then release both the `Ctrl` and the `l` keys, and quickly type the `j` key (OS X users: replace `Ctrl` with `Cmd`). To wrap the selected text in an `\emph{}` command, use `Ctrl-l, Ctrl-e`: that is, hit

`Ctrl-l`, release both keys, then hit `Ctrl-e` (again, OS X users hit `Cmd-l` and then `Cmd-e`).

`Ctrl-l` (`Cmd-l` on OS X) is the standard ST keybinding for "expand selection to line"; this is **remapped** to `Ctrl-l,Ctrl-l` (`Cmd-l,Cmd-l` on OS X). This is the *only* standard ST keybinding that is affected by the plugin—an advantage of new-style keybindings.

Most plugin facilities are invoked using sequences of 2 keys or key combinations, as in the examples just given. A few use sequences of 3 keys or key combinations.

Henceforth, I will write `C-` to mean `Ctrl-` for Linux or Windows, and `Cmd-` for OS X. You know your platform, so you know what you should use. In a few places, to avoid ambiguities, I will spell out which key I mean.

7.1 Compiling LaTeX files

Keybinding: `C-b` (standard ST keybinding)

LaTeXTools offers a fully customizable build process. This section describes the default process, also called "traditional" because it is the same (with minor tweaks) as the one used in previous releases. However, see below for how to customize the build process.

The default ST Build command takes care of the following:

- It saves the current file
- It invokes the tex build command (`texify` for MikTeX; `latexmk` for TeXlive and MacTeX).
- It parses the tex log file and lists all errors, warnings and, if enabled, bad boxes in an output panel at the bottom of the ST window: click on any error/warning/bad boxes to jump to the corresponding line in the text, or use the ST-standard Next Error/Previous Error commands.
- It invokes the PDF viewer for your platform and performs a forward search: that is, it displays the PDF page where the text corresponding to the current cursor position is located.

7.2 Selecting Build Variant

Keybinding: `C-shift-b` (standard ST3 keybinding)

LaTeXTools offers a range of build variants to select standard build options. These variants can be used to customize the options passed to the LaTeXTools builder, so that you don't need a project file or to use any of the `%!TEX` directives to change, e.g., the build system used. Variants are provided for the supported builders and for the supported programs.

In addition, custom Sublime build files can be created to add your own variants to standard LaTeXTools commands. For more on this, see the section on Sublime Build Files.

Note: The settings provided by build variants *override* settings specified in the file itself or in your settings. This means, for example, if you select a build

variant that changes the program, `%!TEX program` directives or `program` settings will be ignored. If you want to return LaTeXTools back to its default behavior, please select the **LaTeX** build variant.

7.3 Toggling window focus following a build

Keybinding: `C-l,t,f` (yes, this means `C-l`, then `t`, then `f`)

By default, after compilation, the focus stays on the ST window. This is convenient if you like to work with the editor and PDF viewer window open side by side, and just glance at the PDF output to make sure that all is OK. If however the editor and viewer windows overlap (e.g. if you have a small screen), you may prefer the viewer window to get the focus (i.e. become the foremost window) after compiling. To this end, you can use the `toggle_focus` command to change this behavior. The first time you invoke this command, the focus will shift to the viewer window after compiling the current file; if you invoke the command again, the post-compilation focus reverts to the editor window. Every time you invoke `toggle_focus`, a message will appear in the status bar.

You can change the default focus behavior via the `keep_focus` option: see the "Settings" section below.

7.4 Toggling PDF syncing (forward search) following a build

Keybinding: `C-l,t,s`

By default, after compilation, LaTeXTools performs a 'forward search' so that the PDF viewer shows the point in the PDF file corresponding to the current cursor position in ST (by the way, you can trigger a forward search at any other time, not just when you are compiling: see below). If for whatever reason you don't like this behavior, you can turn it off using the `toggle_fwdsync` command. As for `toggle_focus`, a message will appear in the status bar to reflect this.

You can also change the default sync behavior via the `forward_sync` option: see the "Settings" section below.

7.5 Checking the status of toggles and defaults

Keybinding: `C-l,t,?`

This opens a quick panel displaying the current toggle values and their corresponding default settings. Selecting an entry in the quick panel will toggle the value (turning the feature on or off).

7.6 Removing temporary files from build

Keybinding: `C-l,backspace`

This deletes all temporary files from a previous build (the PDF file is kept). Subfolders are traversed recursively. This command also clears the LaTeXTools cache.

Two settings allow you to fine-tune the behavior of this command. `temp_files_exts` allows you to specify which file extensions should be considered temporary, and hence deleted. `temp_files_ignored_folders` allows you to specify folders that should not be traversed. A good example are `.git` folders, for people who use git for version control.

Note: If you use any of the special values with the output directory or auxiliary directory feature, the above is ignored, and the entire directory is simply deleted. If you are using the auxiliary directory feature *without* using an output directory, the auxiliary directory will be cleared and the normal process will be run.

7.7 Clearing the cache

Keybinding: `C-l,C-d,C-c`

This clears the LaTeXTools cache. It is useful if the LaTeXTools cache information gets too out of date, but you want to maintain the LaTeX build files, such as `.aux`.

7.8 Show the build panel

Keybinding: `shift+escape`

This will show the LaTeXTools build panel, including any messages from the previous build.

7.9 Forward and Inverse Search

Keybinding: `C-l,j` (for forward search; inverse search depends on the previewer)

When working in an ST view on a TeX document, `C-l,j` will display the PDF page where the text corresponding to the current cursor position is located; this is called a "forward search". The focus is controlled by the `C-l,t,f` toggle setting and the `keep_focus` option.

If you are viewing a PDF file, then hitting `CMD+Shift+Click` in Skim (OSX), double-clicking in Sumatra (Windows), or hitting `Ctrl+click` in Evince (Linux) will bring you to the location in the source tex file corresponding to the PDF text you clicked on. This is called "inverse search".

To open a PDF file without performing a forward search, use `C-l,v`.

For support of forward and inverse search in other viewers, see the viewer section below.

7.10 References and Citations

Keybinding: *autotriggered* by default (see below). Otherwise, `C-l,x` for 'cross-reference,' `C-l,C-f`, or `C-l,C-alt-f` (via the Fill Helper facility: see below). These are fully equivalent ways of invoking ref/cite completions.

The basic idea is to help you insert labels in `\ref{}` commands and bibtex keys in `\cite{}` commands. The appropriate key combination shows a list of available labels or keys, and you can easily select the appropriate one. Full filtering facilities are provided.

Notes:

1. In order to find all applicable labels and bibtex keys, the plugin looks at the **saved** file. So, if you invoke this command and do not see the label or key you just entered, perhaps you haven't saved the file.
2. Only bibliographies in external `.bib` files are supported: no `\bibitem...`. Sorry. It's hard as it is.
3. Multi-file documents are fully supported.

Now for the details. (Many of these features were contributed by Wes Campaigne and jlewegie, whom I thank profusely.)

By default, as soon as you type, for example, `\ref{` or `\cite{`, a quick panel is shown (this is the fancy drop-down list ST displays at the top of the screen), listing, respectively, all the labels in your files, or all the entries in the bibliographies you reference your file(s) using the `\bibliography{}` command. This is the default *auto-trigger* behavior, and it can be a big time saver. You can, however, turn it off, either temporarily using a toggle, or permanently by way of preference settings: see below. Once the quick panel is shown, you can narrow down the entries shown by typing a few characters. As with any ST quick panel, what you type will be fuzzy-matched against the label names or, for citations, the content of the first displayed line in each entry (by default, the author names, year of publication, short title and citation key: see below). This is *wildly* convenient, and one of the best ST features: try it!

If auto-triggering is off, when you type e.g. `\ref{`, ST helpfully provides the closing brace, leaving your cursor between the two braces. Now, you need to type `C-l,x` to get the quick panel showing all labels in the current file. You can also type e.g. `\ref{aa` [again, the closing brace is provided by ST], then `C-l, x`, and LaTeXTools will show a list of labels that fuzzy-match the string `aa`.

In either case, you then select the label you want, hit Return, and LaTeXTools inserts the **full ref command**, as in `\ref{my-label}`. The LaTeX command `\eqref` works the same way. Citations from bibtex files are also supported in a similar way. Use `\cite{}`, `\citet{}`, `\citeyear{}` etc.

One often needs to enter **multiple citations**, as e.g. in `\cite{paper1,paper2}`. This is easy to do: either cite the first paper, e.g. `\cite{paper1}` and then, *with your cursor immediately before the right brace*, type a comma (`,`). Again, the default auto-trigger behavior is that the quick panel with appear, and you can select the second paper. If auto-trigger is off, then you enter the comma, then

use the shortcut `C-l,x` to bring up the quick panel (note: you *must* add the comma before invoking the shortcut, or you won't get the intended result). Of course, you can enter as many citations as you want.

The display of bibliographic entries is *customizable*. There is a setting, `cite-panel-format`, that controls exactly what to display in each of the two lines each entry gets in the citation quick panel. Options include author, title, short title, year, bibtex key, and journal. This is useful because people may prefer to use different strategies to refer to papers—author-year, short title-year, bibtex key (!), etc. Since only the first line in each quick panel entry is searchable, how you present the information matters. The default should be useful for most people; if you wish to change the format, check the `LaTeXTools.sublime-settings` file for detailed information. (As usual, copy that file to the `User` directory and edit your copy, not the original).

Thanks to recent contributed code, **multi-file documents** are *fully supported*. LaTeXTools looks for references, as well as `\bibliography{}` commands, in the root file and in all recursively included files. Please see the information on **Multi-file documents** in the section on **General Features** for details on how to setup multi-file documents.

LaTeXTools now also looks `\addbibresource{}` commands, which provides basic compatibility with biblatex.

7.10.1 Old-style, deprecated functionality

For now, completions are also injected into the standard ST autocompletion system. Thus, if you hit `Ctrl-space` immediately after typing, e.g., `\ref{}`, you get a drop-down menu at the current cursor position (not a quick-panel) showing all labels in your document. However, the width of this menu is OK for (most) labels, but not really for paper titles. In other words, it is workable for references, but not really for citations. Furthermore, there are other limitations dictated by the ST autocompletion system. So, this is **deprecated**, and I encourage you to use auto-trigger mode or the `C-l,x` or `C-l,C-f` keybindings instead.

7.11 Forcing Citations and References

Keybinding: `C-l,alt-x,c` (citations) or `C-l,alt-x,r` (refs)

In some cases, it may be desirable to forcibly insert a citation key or label, i.e., if LaTeXTools does not automatically understand the command you are using. In such circumstances, you can use these keybindings to forcibly insert either a citation or reference at the cursor position. Note that the current word won't be overridden and any open brackets will not be completed if either of these options are used.

7.12 Toggle auto trigger mode on/off

Keybinding: `C-l,t,a,r` for references; `C-l,t,a,c` for citations; `C-l,t,a,f` for input files; `C-l,t,a,e` for environments; `C-l,t,a,b` for smart brackets

These toggles work just like the sync and focus toggles above. Indeed, `C-l,t,?` will now also display the status of the auto trigger toggles. Check the status bar for feedback (i.e. to see what the current state of the toggle is), but remember the message stays on for only a few seconds. `C-l,t,?` is your friend.

7.13 Fill Helper: filling in package and file names automatically

Keybinding: *autotriggered* by default (see below). Otherwise, `C-l,C-f` or `C-l,C-alt-f` (see below).

Thanks to the amazing work by users btstream and Ian Bacher, LaTeXTools now offers a list of available files and packages when using commands such as `\usepackage`, `\include`, `\includegraphics`, `\includesvg` and `\input`. Assuming autocompletion is toggled on (the default):

- when you type `\usepackage{`, a list of available package is displayed in the ST drop-down menu. Pick the one you need, and it will be inserted in your file, with a closing brace.
- when you type any of the file-related input commands, a list of files in the current directory is displayed (suitably filtered, so graphics files are displayed for `\includegraphics`).

To toggle autocompletion on or off, use the `fill_auto_trigger` setting, or the `C-l,t,a,f` toggle.

In order for package autocomplete to work, you need to create a cache first. You can do it using the Command Palette: select `LaTeXTools: Build cache for LaTeX packages`.

The `C-l,C-f` keyboard shortcut also works for `\ref` and `\cite` completion. Basically, wherever you can use `C-l,x`, you can also use `C-l,C-f`.

The `C-l,C-alt-f` keyboard shortcut is identical to the `C-l,C-f` shortcut, except that it ensures that the current word that the cursor is within is replaced. This is useful for, e.g., switching one citation for another or one label for another.

7.14 Jumping to sections and labels

Keybinding: `C-r` (standard ST keybinding)

The LaTeXtools plugin integrates with the awesome ST "Goto Anything" facility. Hit `C-r`to get a list of all section headings, and all labels. You can filter by typing a few initial letters. Note that section headings are preceded by the letter "S", and labels by "L"; so, if you only want section headings, type "S" when the drop-down list appears.

Selecting any entry in the list will take you to the corresponding place in the text.

7.15 Jump to Anywhere

Keybinding: `C-l, C-j` or `C-l, C-o` (see below)

Mousebinding: `ctrl-alt-leftclick` (Windows) / `super-leftclick` Linux) / `ctrl-super-leftclick` (OSX)

Mousebinding (With SublimeCodeIntel): `alt-leftclick` (Windows) / `super-leftclick` Linux) / `ctrl-leftclick` (OSX)

This is an IDE-like mouse navigation, which executes a jump depending on the context around the cursor. It is easy to use and intuitive. Just click with the mouse on a command while pressing the modifier key. The corresponding jump will be executed. Supported jump types are:

- Jump to referenced labels (e.g. `\ref`)
- Jump to citation entries in bibliography files (e.g. `\cite`)
- Open included files (e.g. `\input` or `\include`)
- Open root file from `%!TEX root =...` magic comment
- Open bibliographies (e.g. `\bibliography` or `\addbibresource`)
- Open included graphics with a specified program (e.g. `\includegraphics`)
- Open the documentation of used packages (e.g. `\usepackage`)
- Jump to self-defined command definition, i.e. jump to the `\newcommand` in which the command was defined

7.15.1 SublimeCodeIntel Integration

If you use SublimeCodeIntel you recognize the alternative mouse-bindings and it does not work out of the box. Just open the command palette and run the command `LaTeXTools: Create Mousemap in User folder`. This will create a mouse-map in the user folder or modify the existing one to add the mouse-binding with the same modifiers as SublimeCodeIntel. This mouse-binding has a `fallback_command` command as argument. This command will be executed if the command in called outside a LaTeX document.

7.15.2 Jumping to included files

To open a file included using, e.g., `\input` or `\include` or a bibliography, simply click while holding down the modifier key or press `C-l, C-j`. Sublime will open the included file.

There is an additional keybinding `C-l, C-o` to jump to included files. If the file already exists, this behaves just like `C-l, C-j`. However, if the file does not exist, it will also create the missing file and, if a `tex` file, will add the magic root entry (`%!TEX root=`) to the new file. This can be used to easily create files or to open already existing files.

7.15.3 Image files

To open an image, which is included with `\includegraphics` just click on the command while pressing the modifier key or press `C-l, C-j`.

The program to open the image can be configured in the LaTeXTools settings in the `open_image_command` setting.

The following sub-settings are provided:

- `image_types`: a list of the image file types used in the `\includegraphics` command. This list is also used in the Fill Helper and to determine missing extensions to open images. When opening an image the `image_types`-list will be matched from left to right.
- `open_image_command`: the command/program to open an image used in the `\includegraphics` command. This commands can be configured OS-specific. For each OS you can create a list, which will be searched top-down for the matching extension. Each entry in the list has a `command` and `extension` field. The command is a string and will be executed with the file path appended, if the extension matches the extension of the file. You can optionally use `$file` inside the string to insert the file path at an arbitrary position. The `extension` can either be a string or a list of string. If it is missing, the command will be executed for every file type.

7.15.4 Packages

If you use the command while the cursor is inside a `\usepackage` command, the documentation for the corresponding package will be opened in your default PDF viewer.

7.16 LaTeX commands and environments

Keybindings: `C-l,c` for commands and `C-l,e` for environments

To insert a LaTeX command such as `\color{}` or similar, type the command without backslash (i.e. `color`), then hit `C-l,c`. This will replace e.g. `color` with `\color{}` and place the cursor between the braces. Type the argument of the command, then hit Tab to exit the braces.

Similarly, typing `C-l,e` gives you an environment: e.g. `test` becomes

```
\begin{test}

\end{test}
```

and the cursor is placed inside the environment thus created. Again, Tab exits the environment.

Note that all these commands are undoable: thus, if e.g. you accidentally hit `C-l,c` but you really meant `C-l,e`, a quick `C-z`, followed by `C-l,e`, will fix things.

7.17 Wrapping existing text in commands and environments

Keybindings: `C-l,C-c`, `C-l, C-n`, etc.

The tab-triggered functionality just described is mostly useful if you are creating a command or environment from scratch. However, you sometimes have existing

text, and just want to apply some formatting to it via a LaTeX command or environment, such as `\emph` or `\begin{theorem}...\end{theorem}`.

LaTeXTools' wrapping facility helps you in just these circumstances. All commands below are activated via a key binding, and *require some text to be selected first*. Also, as a mnemonic aid, *all wrapping commands involve typing `C-l,C-something` (which you can achieve by just holding the `C-` key down after typing `l`).

- `C-l,C-c` wraps the selected text in a LaTeX command structure. If the currently selected text is `blah`, you get `\cmd{blah}`, and the letters `cmd` are highlighted. Replace them with whatever you want, then hit Tab: the cursor will move to the end of the command.
- `C-l,C-e` gives you `\emph{blah}`, and the cursor moves to the end of the command.
- `C-l,C-b` gives you `\textbf{blah}`
- `C-l,C-u` gives you `\underline{blah}`
- `C-l,C-t` gives you `\texttt{blah}`
- `C-l,C-n` wraps the selected text in a LaTeX environment structure. You get `\begin{env}`,`blah`, `\end{env}` on three separate lines, with `env` selected. Change `env` to whatever environment you want, then hit Tab to move to the end of the environment.

These commands also work if there is no selection. In this case, they try to do the right thing; for example, `C-l,C-e` gives `\emph{}` with the cursor between the curly braces.

You can also *change the current environment* using the `C-l,C-Shift-n` shortcut. Note well how this works. First, the cursor must be inside the environment you are interested in. Second, the command selects the environment name in the `\begin{env}` command and also in the `\end{env}` command (using ST's multiple-selection support). This way you can rename the environment as needed. *Remember to exit multiple-selection mode* when you are done by pressing the `ESC` key.

7.18 Word Count

Keybinding: `C-l,w`

This uses TeXcount to generate a word count for the current document which is displayed in a quick panel. If you don't have the `TeXcount`, you will simply get an error message. Word counts in LaTeX documents can be quite finicky, and its worth reviewing the TeXcount documentation to ensure your document is setup to generate as accurate a word-count as possible. The counts returned are those reported by: `texcount -total -merge <main_file.tex>`.

The `word_count_sub_level` setting can be tweaked to display subcounts by chapter, section, etc. See the Settings below.

8 Completions

8.1 Command completion, snippets, etc.

By default, ST provides a number of snippets for LaTeX editing; the LaTeXTools plugin adds a few more. You can see what they are, and experiment, by selecting **Tools|Snippets|LaTeX** and **Tools|Snippets|LaTeXTools** from the menu.

In addition, the LaTeXTools plugin provides useful completions for both regular and math text; check out files `LaTeX.sublime-completions` and `LaTeX math.sublime-completions` in the LaTeXTools directory for details. Some of these are semi-intelligent: i.e. `bf` expands to `\textbf{}` if you are typing text, and to `\mathbf{}` if you are in math mode. Others allow you to cycle among different completions: e.g. `f` in math mode expands to `\phi` first, but if you hit Tab again you get `\varphi`; if you hit Tab a third time, you get back `\phi`.

8.2 LaTeX-cwl support

LaTeXTools provides support for the `LaTeX-cwl` autocompletion word lists. If the package is installed, support is automatically enabled. In addition, support will be enabled if any custom cwl files are installed in the `Packages/User/cwl` directory.

By default, as soon as one starts typing a command, e.g., `\te`, a popup is shown displaying possible completions, e.g. including `\textit` and the like.

The following settings are provided to control LaTeXTools cwl behavior.

- `cwl_list`: a list of `cwl` files to load
- `cwl_autoload`: controls loading completions based on packages in the current document *in addition* to those specified in the `cwl_list`. Defaults to `true`, so you only need to set this if you want to *disable* this behavior.
- `command_completion`: when to show that cwl completion popup. The possible values are:
  - `prefixed` (default): show completions only if the current word is prefixed with a `\`
  - `always`: always show cwl completions
  - `never`: never display the popup
- `env_auto_trigger`: if `true`, autocomplete environment names upon typing `\begin{` or `\end{` (default: `false`)

9 Settings

LaTeXTools supports user-defined settings. The settings file is called `LaTeXTools.sublime-settings`. A default version resides in the LaTeXTools plugin directory and **must not be edited**. This contains default settings that will work in many cases, for standard configurations of TeX distros and PDF viewers. You can however create another settings file in your `User` directory; again, the file must be named `LaTeXTools.sublime-settings`.

You can create and edit such a file manually. It is a standard Sublime Text JSON file; the settings currently honored by LaTeXTools are listed below. However, the *simplest way to create a settings file* is to open the **Preferences | Package Settings | LaTeXTools** submenu and select the **Settings - User** option. If you do not currently have a `LaTeXTools.sublime-settings` settings file in your `User` directory (e.g., if you are installing LaTeXTools for the first time), you will be given the option to create one. The newly created settings file will be an exact copy of the default one, and will open in a tab for you to customize.

If you *do* already have an existing `LaTeXTools.sublime-settings` file in your `User` directory, the `Settings - User` option will open that file in a tab for you to further customize. Similarly, the `Settings - Default` option will open the default settings file in a tab, in *read-only mode*. This may be useful for you to copy from, or if you want to see what other options may be available without consulting this README file.

If at any time you wish to erase your customizations and start afresh, you can simply delete the `LaTeXTools.sublime-settings` file in your `User` directory. (Again, *warning*: do *not* touch the settings file in the LaTeXTools plugin directory!) Alternatively, from the **Preferences | Package Settings | LaTeXTools** submenu, or from the Command Palette, you can choose **Reset user settings to default**. This will delete any existing settings file in `User` and create a copy of the default one. This will *remove all your customizations.*

(Historical note: This is no longer relevant in 2016, but just for the record, if you have a pre-2014, old-style settings file, this option will import it).

Warning: in general, tweaking options can cause breakage. For instance, if on Linux you change the default `python` setting (empty by default) to a non-existent binary, forward and inverse search will stop working. With great power comes great responsibility! If you think you have found a bug, *delete your settings file in the `User` directory, or use the `Reset user settings to default` command before reporting it!* Thanks :-)

The following options are currently available (defaults in parentheses):

9.1 General Settings

- `cite_auto_trigger` (`true`): if `true`, typing e.g. `\cite{` brings up the citation completion quick panel, without the need to type `C-l,x`. If `false`, you must explicitly type `C-l,x`.
- `ref_auto_trigger` (`true`): ditto, but for `\ref{` and similar reference commands
- `fill_auto_trigger` (`true`): ditto, but for package and file inclusion commands (see Fill Helper feature above)
- `env_auto_trigger` (`true`): ditto, but for environment completions
- `cwl_autoload` (`true`): whether to load cwl completions based on packages (see the LaTeX-cwl feature)
- `cwl_completion` (`prefixed`): when to activate the cwl completion poput (see LaTeX-cwl feature above)
- `cwl_list` (`["latex-document.cwl", "tex.cwl", "latex-dev", "latex-209.cwl", "latex-l2tabu.cwl", "latex-mathsymbols.cwl"]`): list of cwl files to load

- `keep_focus` (`true`): if `true`, after compiling a tex file, ST retains the focus; if `false`, the PDF viewer gets the focus. Also note that you can *temporarily* toggle this behavior with `C-l,t,f`. **Note**: If you are on either Windows or Linux you may need to adjust the `sublime_executable` setting for this to work properly. See the **Platform settings** below. This can also be overridden via a key-binding by passing a `keep_focus` argument to `jump_to_pdf`.
- `forward_sync` (`true`): if `true`, after compiling a tex file, the PDF viewer is asked to sync to the position corresponding to the current cursor location in ST. You can also *temporarily* toggle this behavior with `C-l,t,s`. This can also be overridden via a key-binding by passing a `forward_sync` argument to `jump_to_pdf`.
- `build_finished_message_length` (`2.0`): the number of seconds to display the notification about the completion of the build in the status bar.ile extensions to be considered temporary, and hence deleted using the `C-l, backspace` command.
- `temp_files_ignored_folders`: subdirectories to skip when deleting temp files.
- `tex_file_exts` (`['.tex']`): a list of extensions that should be considered TeX documents. Any extensions in this list will be treated exactly the same as `.tex` files. See the section on Support for non-`.tex` files.
- `latextools_set_syntax` (`true`): if `true` LaTeXTools will automatically set the syntax to `LaTeX` when opening or saving any file with an extension in the `tex_file_exts` list.
- `use_biblatex`: (`false`): if `true` LaTeXTools will use BibLaTeX defaults for editing `.bib` files. If `false`, LaTeXTools will use BibTeX defaults. See the section on Support for Editing Bibliographies for details.
- `tex_spellcheck_paths` (`{}`): A mapping from the locales to the paths of the dictionaries. See the section Spell-checking.
- `word_count_sub_level` (`"none"`): controls the level at which subcounts of words can be generated. Valid values are: `"none"`, `"part"`, `"chapter"`, and `"section"`.

9.2 Platform-Specific Settings

This section refers to setting that can be found in a platform-specific block for each platform, i.e., `"osx"`, `"windows"`, or `"linux"`.

9.2.1 All Platforms

- `texpath` (varies): the path to TeX & friends

9.2.2 Windows

- `distro` (`miktex`): either `miktex` or `texlive`, depending on your TeX distribution
- `sumatra` (`""`): leave blank or omit if the SumatraPDF executable is in your `PATH` and is called `SumatraPDF.exe`, as in a default installation; otherwise, specify the *full path and file name* of the SumatraPDF executable.
- `sublime_executable` (`""`): this is used if `keep_focus` is set to true and the path to your sublime_text executable cannot be discovered automatically. It should point to the full path to your executable `sublime_text.exe`.

- `keep_focus_delay` (`0.5`): this is used if `keep_focus` is set to true. It controls how long (in seconds) the delay is between the completion of the `jump_to_pdf` command and the attempt to refocus on Sublime Text. This may need to be adjusted depending on your machine or configuration.

9.2.3 Linux

- `python` (`""`, i.e. empty string): name of the Python executable. This is useful if you've installed Python in a non-standard location or want to ensure that LaTeXTools uses a particular Python version. Note that the Python interpreter you select must have the DBus bindings installed.
- `sublime` (`sublime-text`): name of the ST executable. Ubuntu supports both `sublime-text` and `subl`; other distros may vary.
- `sync_wait` (1.5): when you ask LaTeXTools to do a forward search, and the PDF file is not yet open (for example, right after compiling a tex file for the first time), LaTeXTools first launches evince, then waits a bit for it to come up, and then it performs the forward search. This parameter controls how long LaTeXTools should wait. If you notice that your machine opens the PDF, then sits there doing nothing, and finally performs the search, you can decrease this value to 1.0 or 0.5; if instead the PDF file comes up but the forward search does not seem to happen, increase it to 2.0.
- `sublime_executable` (`""`): this is used if `keep_focus` is set to true and the path to your sublime_text executable cannot be discovered automatically. It should point to the full path to your executable `sublime_text`.
- `keep_focus_delay` (`0.5`): this is used if `keep_focus` is set to true. It controls how long (in seconds) the delay is between the completion of the `jump_to_pdf` command and the attempt to refocus on Sublime Text. This may need to be adjusted depending on your machine or configuration.

9.3 Output and Auxiliary Directory settings

- `aux_directory` (`""`): specifies the auxiliary directory to store any auxiliary files generated during a LaTeX build. Note that the auxiliary directory option is only useful if you are using MiKTeX. Path can be specified using either an absolute path or a relative path. If `aux_directory` is set from the project file, a relative path will be interpreted as relative to the project file. If it is set in the settings file, it will be interpreted relative to the main tex file. In addition, the following special values are honored:
- `<<temp>>`: uses a temporary directory in the system temp directory instead of a specified path; this directory will be unique to each main file, but does not persist across restarts.
- `<<cache>>`: uses the ST cache directory (or a suitable directory on ST2) to store the output files; unlike the `<<temp>>` option, this directory can persist across restarts.
- `<<project>>`: uses a sub-directory in the same folder as the main tex file with what should be a unique name; note, this is probably not all that useful and you're better off using one of the other two options or a named relative path

- `output_directory` (`""`): specifies the output directory to store any file generated during a LaTeX build. Path can be specified using either an absolute path or a relative path. If `output_directory` is set from the project file, a relative path will be interpreted as relative to the project file. If it is set in the settings file, it will be interpreted relative to the main tex file. In addition, output_directory honors the same special values as `auxiliary_directory`.
- `jobname` (`""`): specifies the jobname to use for the build, corresponding to the pdflatex `--jobname` argument.
- `copy_output_on_build` (`true`): if `true` and you are using an `output_directory`, either set via the setting or the `%!TEX` directive, this instructs LaTeXTools to copy to resulting pdf to the same folder as the main tex file. If you are not using `output_directory` or it is set to `false`, it does nothing. If it is a list of extensions, it will copy each file with the same name as your main tex file and the given extension to the same folder as your main tex file. This is useful for copying, e.g., .synctex.gz or .log files.

9.4 Builder Settings

Note: for the time being, you will need to refer to the `LaTeXTools.sublime-settings` file for detailed explanations. Also, since the new build system is meant to be fully customizable, if you use a third-party builder (which hopefully will become available!), you need to refer to its documentation.

- `builder`: the builder you want to use. Leave blank (`""`) or set to `"default"` or `"traditional"` for the traditional (`latexmk`/`texify`) behavior. Set to `"basic"` for the basic builder that supports output and auxiliary directories on MiKTeX.
- `builder_path`: builders can reside anywhere Sublime Text can access. Specify a path *relative to the Sublime text Packages directory*. In particular, `User` is a good choice. If you use a third-party builder, specify the builder-provided directory.
- `builder-settings`: these are builder-specific settings. For the `default`/`traditional` builder, the following settings are useful:
  - `program` (unset): one of `pdflatex` (the default), `xelatex` or `lualatex`. This selects the TeX engine.
  - `command` (unset): the precise `latexmk` or `texify` command to be invoked. This must be a list of strings. The defaults (hardcoded, not shown in the settings file) are:
    - (TeXLive): `["latexmk", "-cd", "-e", "-f", "-%E", "-interaction=nonstopmode", "-synctex=1"]`
    - (MiKTeX): `["texify", "-b", "-p", "--engine=%E", "--tex-option=\"--synctex=1\""]`
  - `options` (unset): allows you to specify a TeX option, such as `--shell-escape`. This must be a tuple: that is, use `options: ["--shell-escape"]`
  - `env` (unset): a dictionary of key-values corresponding to environment variables that should be set for the environment the build is run in. Note that `env`, if it is set, must be set at the platform-specific level, e.g., under the `osx`, `windows`, or `linux` keys. This is useful for setting, e.g., `TEXINPUTS`.

  - In addition, there can be platform-specific settings. An important one for Windows is `distro`, which must be set to either `miktex` or `texlive`.

9.5 Build Panel Settings

- `highlight_build_panel` (`true`): if `true` the build panel will have a syntax applied to highlight any errors and warnings. Otherwise, the standard output panel configuration will be used.
- `hide_build_panel` (`"never"`): controls whether or not to hide the build panel after a build is finished. Possible values:
  - `"always"` - hide the panel even if the build failed
  - `"no_errors"` - only hide the panel if the build was successful even with warnings
  - `"no_warnings"` - only hide the panel if no warnings occur
  - `"no_badboxes"` - only hide the panel if no warnings or badbox messages occur; this only differs from `no_warnings` if `display_bad_boxes` is set to `true`.
  - `"never"` - never hide the build panel
    Any other value will be interpretted as the default.
- `display_bad_boxes` (`false`): if `true` LaTeXTools will display any bad boxes encountered after a build. Note that this is disabled by default.

9.6 Viewer settings

- `viewer` (`""`): the viewer you want to use. Leave blank (`""`) or set to `"default "`for the platform-specific viewer. Can also be set to `"preview"` if you want to use Preview on OS X, `"okular"` if you want to use Okular on Linux, `"zathura"` is you want to use Zathura on Linux, or `"command"` to run arbitrary commands. For details on the `"command"` option, see the section on the Command Viewer.
- `viewer_settings`: these are viewer-specific settings. Please see the section on Viewers or the documentation on Alternate Viewers for details of what should be set here.
- `open_pdf_on_build` (`true`): Controls whether LaTeXTools will automatically open the configured PDF viewer on a successful build. If set to `false`, the PDF viewer will only be launched if explicitly requested using `C-l,v` or `C-l,j`.

9.7 Bibliographic references settings

- `bibliography` (`"traditional_bibliography"`): specifies the bibliography plugin to use to handle extracting entries from a bibliography.
- `cite-panel-format` and `cite_autocomplete_format`: see the section on ref/cite completion, and the comments in `LaTeXTools.sublime-settings`

9.8 Cache settings

- `hide_local_cache` (`true`): Whether the local cache should be hidden in the sublime cache path (`true`) or in the same directory as the root file (`false`). See the section LaTeXTools Cache.
- `local_cache_life_span` (`30 m`): The lifespan of the local cache, specified in the format `" d x h X m X s"` where `X` is a natural number `s` stands for seconds, `m` for minutes, `h` for hours, and `d` for days. Missing fields will be treated as 0 and white-spaces are optional. Hence you can write `"1 h 30 m"` to refresh the cached data every one and a half hours. If you use `"infinite"` the cache will not be invalidated automatically. A lower lifespan will produce results, which are more up to date. However it requires more recalculations and might decrease the performance. See the section LaTeXTools Cache.

9.9 Project-Specific Settings

The above settings can be overridden on a project-specific basis if you are using Sublime Text's project system. To override these settings, simply create a `"settings"` section in your project file. The structure and format is the same as for the `LaTeXTools.sublime-settings` file. Here is an example:

```
{
    ...<folder-related options here>...

    "settings" : {
        "TEXroot": "main.tex",
        "tex_file_exts": [".tex", ".tikz"],
        "builder_settings": {
            "program": "xelatex",
            "options": "--shell-escape"
        }
    }
}
```

This sets `main.tex` as the master tex file (assuming a multi-file project), and allows `.tikz` files to be recognized as tex files, but only for the current project. Furthermore (using the default, i.e., traditional builder), it forces the use of `xelatex` instead of the default `pdflatex`, and also adds the `--shell-escape` option—again, for the current project only.

Note: tweaking settings on a project-specific level can lead to even more subtle issues. If you notice a bug, in addition to resetting your `LaTeXTools.sublime-settings` file, you should remove at LaTeXTools settings from your project file.

10 Alternative Builders

10.1 Basic Builder

The basic builder is a simple, straight-forward build system. that simply runs the configured build engine (pdflatex, xelatex, or lualatex) and bibtex or biber if necessary. It can also be configured to support bibtex8 through the `bibtex`

builder setting. In addition, it supports the TeX Options feature, the output and auxiliary directory features and the Jobname feature. It has been included because the default builder on MiKTeX, `texify` cannot be easily coerced to support biber or any of the other features supported by the basic builder. Note, however, that unlike `texify`, the basic builder does **not** support `makeindex` and friends (patches are welcome!).

You can use the basic builder by changing the `builder` setting to `"basic"`. It will read the same settings as the traditional builder.

10.2 Script Builder

LaTeXTools now supports the long-awaited script builder. It has two primary goals: first, to support customization of simple build workflows and second, to enable LaTeXTools to integrate with external build systems in some fashion.

Note that the script builder should be considered an advanced feature. Unlike the "traditional" builder it is not designed to "just work," and is not recommend for those new to using TeX and friends. You are responsible for making sure your setup works. Please read this section carefully before using the script builder.

For the most part, the script builder works as described in the Compiling LaTeX files section *except that* instead of invoking either `texify` or `latexmk`, it invokes a user-defined series of commands. Note that although the Script Builder supports **Multi-file documents**, it does not support either the engine selection or passing other options via the `%!TEX` macros.

The script builder is controlled through two settings in the *platform-specific* part of the `builder_settings` section of `LaTeXTools.sublime-settings`, or of the current project file (if any):

- `script_commands` — the command or list of commands to run. This setting **must** have a value or you will get an error message.
- `env` — a dictionary defining any environment variables to be set for the environment the command is run in.

The `script_commands` setting should be either a string or a list. If it is a string, it represents a single command to be executed. If it is a list, it should be either a list of strings representing single commands or a list of lists, though the two may be mixed. For example:

```
{
    "builder_settings": {
        "osx": {
            "script_commands":
                "pdflatex -synctex=1 -interaction=nonstopmode"
        }
    }
}
```

Will simply run `pdflatex` against the master document, as will:

```
{
    "builder_settings": {
        "osx": {
            "script_commands":
```

```
                ["pdflatex -synctex=1 -interaction=nonstopmode"]
        }
    }
}
```

Or:

```
{
    "builder_settings": {
        "osx": {
            "script_commands":
                [["pdflatex", "-synctex=1 -interaction=nonstopmode"]]
        }
    }
}
```

More interestingly, the main list can be used to supply a series of commands. For example, to use the simple pdflatex -> bibtex -> pdflatex -> pdflatex series, you can use the following settings:

```
{
    "builder_settings": {
        "osx": {
            "script_commands": [
                "pdflatex -synctex=1 -interaction=nonstopmode",
                "bibtex",
                "pdflatex -synctex=1 -interaction=nonstopmode",
                "pdflatex -synctex=1 -interaction=nonstopmode"
            ]
        }
    }
}
```

Note, however, that the script builder is quite unintelligent in handling such cases. It will not note any failures nor only execute the rest of the sequence if required. It will simply continue to execute commands until it hits the end of the chain of commands. This means, in the above example, it will run `bibtex` regardless of whether there are any citations.

It is especially important to ensure that, in case of errors, TeX and friends do not stop for user input. For example, if you use `pdflatex` on either TeXLive or MikTeX, pass the `-interaction=nonstopmode` option.

Each command can use the following variables which will be expanded before it is executed:

Variable	Description
`$file`	The full path to the main file, e.g., *C:\Files\document.tex*
`$file_name`	The name of the main file, e.g., *document.tex*
`$file_ext`	The extension portion of the main file, e.g., *tex*
`$file_base_name`	The name portion of the main file without the, e.g., *document*
`$file_path`	The directory of the main file, e.g., *C:\Files*
`$aux_directory`	The auxiliary directory set via a `%!TEX` directive or the settings
`$output_directory`	The output directory set via a `%!TEX` directive or the settings

Variable	Description
`$jobname`	The jobname set via a `%!TEX` directive or the settings

For example:

```
{
    "builder_settings": {
        "osx": {
            "script_commands": [[
                "pdflatex",
                "-synctex=1"
                "-interaction=nonstopmode",
                "$file_base_name"
            ]]
        }
    }
}
```

Note that if none of these variables occur in the command string, the `$file_base_name` will be appended to the end of the command. This may mean that a wrapper script is needed if, for example, using `make`.

Commands are executed in the same path as `$file_path`, i.e. the folder containing the main document. Note, however, on Windows, since commands are launched using `cmd.exe`, you need to be careful if your root document is opened via a UNC path (this doesn't apply if you are simply using a mapped drive). `cmd.exe` doesn't support having the current working directory set to a UNC path and will change the path to `%SYSTEMROOT%`. In such a case, just ensure all the paths you specify are absolute paths and use `pushd` in place of `cd`, as this will create a (temporary) drive mapping.

10.2.1 Supporting output and auxiliary directories

If you are using LaTeXTools output and auxiliary directory behavior there are some caveats to be aware of. First, it is, of course, your responsibility to ensure that the approrpiate variables are passed to the appropriate commands in your script. Second, `pdflatex` and friends do not create output directories as needed. Therefore, at the very least, your script must start with either `"mkdir $output_directory"` (Windows) or `"mkdir -p $output_directory"` and a corresponding command if using a separate `$aux_directory`. Note that if you `\include` (or otherwise attempt anything that will `\@openout` a file in a subfolder), you will need to ensure the subfolder exists. Otherwise, your run of `pdflatex` will fail.

Finally, unlike Biber, bibtex (and bibtex8) does not support an output directory parameter, which can make it difficult to use if you are using the LaTeXTools output directory behavior. The following work-arounds can be used to get BibTeX to do the right thing.

On Windows, run BibTeX like so:

```
cd $aux_directory & set BIBINPUTS=\"$file_path:%BIBINPUTS%\" &
    bibtex $file_base_name
```

And on OS X or Linux, use this:

```
"cd $output_directory; BIBINPUTS=\"$file_path;$BIBINPUTS\" bibtex
    $file_base_name"
```

In either case, these run bibtex *inside* the output / auxiliary directory while making the directory containing your main file available to the `BIBINPUTS` environment variable. Note if you use a custom style file in the same directory, you will need to apply a similar work-around for the `BSTINPUTS` environment variable.

10.2.2 Supporting jobname

If you are using LaTeXTools jobname behaviour, you should be aware that you are responsible for ensure jobname is set in the appropriate context. In particular, a standard build cycle might look something like this:

```
{
    "builder_settings": {
        "osx": {
            "script_commands": [
                [
                    "pdflatex",
                    "-synctex=1"
                    "-interaction=nonstopmode",
                    "-jobname=$jobname",
                    "$file_base_name"
                ],
                [
                    "bibtex",
                    "$jobname"
                ],
                [
                    "pdflatex",
                    "-synctex=1"
                    "-interaction=nonstopmode",
                    "-jobname=$jobname",
                    "$file_base_name"
                ],
                [
                    "pdflatex",
                    "-synctex=1"
                    "-interaction=nonstopmode",
                    "-jobname=$jobname",
                    "$file_base_name"
                ]
            ]
        }
    }
}
```

10.2.3 Caveats

LaTeXTools makes some assumptions that should be adhered to or else things won't work as expected:

- the final product is a PDF which will be written to the output directory or the same directory as the main file and named `$file_base_name.pdf`
- the LaTeX log will be written to the output directory or the same directory as the main file and named `$file_base_name.log`

- if you change the `PATH` in the environment (by using the `env` setting), you need to ensure that the `PATH` is still sane, e.g., that it contains the path for the TeX executables and other command line resources that may be necessary.

In addition, to ensure that forward and backward sync work, you need to ensure that the `-synctex=1` flag is set for your latex command. Again, don't forget the `-interaction=nonstopmode` flag (or whatever is needed for your tex programs not to expect user input in case of error).

Finally, please remember that script commands on Windows are run using `cmd.exe` which means that if your script uses any UNC paths will have to use `pushd` and `popd` to properly map and unmap a network drive.

10.3 Sublime Build Files

LaTeXTools now has some support for custom `.sublime-build` files or builders specified in your project settings. For an overview of `.sublime-build` files in general, please see the Unofficial Documentation (which is generally a great resource about Sublime Text). For more on adding builders to project files, see the relevant section of the Sublime documentation. This section will cover the basics of creating a `.sublime-build` file that works with LaTeXTools.

At a minimum, your `.sublime-build` file must have the following elements:

```
{
    "target": "make_pdf",
    "selector": "text.tex.latex",

    "osx":
        {
            "file_regex": "^(...*?):([0-9]+): ([0-9]*)([^\\.]+)"
        },

    "windows":
        {
            "file_regex": "^((?:.:)?[^:\n\r]*):([0-9]+):?([0-9]+)?:? (.*)$"
        },

    "linux":
        {
            "file_regex": "^(...*?):([0-9]+): ([0-9]*)([^\\.]+)"
        }
}
```

Otherwise, other features may not work as expected. In addition, you can specify the following other parameters:

Parameter	Description
`builder`	Overrides the `builder` setting. May refer to any valid LaTeXTools builder.
`program`	Overrides the `program` setting or `%!TEX program` macro. May be one of `pdflatex`, `xelatex`, or `lualatex`

Parameter	Description
`command`	Overrides the `command` setting, providing the command run by the builder. This is only useful if you use the `traditional` builder. For the format, see the relevant builder setting.
`env`	Overrides the `env` setting. Should be a dictionary similar to `env`, but note that when specified in a `.sublime-build` file, it is not, by default, platform-specific.
`path`	Overrides the `texpath` settings. Note that if you set this, you are responsible for ensuring that the appropriate LaTeX install can still be found.

10.4 Custom Builders

Since the release on March 13, 2014 (v3.1.0), LaTeXTools has had support for custom build systems, in addition to the default build system, called the "traditional" builder. Details on how to customize the traditional builder are documented above. If neither the traditional builder nor the script builder meet your needs you can also create a completely custom builder which should be able to support just about anything you can imagine. Let me know if you are interested in writing a custom builder!

Custom builders are small Python scripts that interact with the LaTeXTools build system. In order to write a basic builder it is a good idea to have some basic familiarity with the Python language. Python aims to be easy to understand, but to get started, you could refer either to the Python tutorial or any of the resources Python suggests for non-programmers or those familiar with other programming languages.

LaTeXTools comes packaged with a small sample builder to demonstrate the basics of the builder system, called `SimpleBuilder` which can be used as a reference for what builders can do.

If you are interested in developing your own builder, please see our page on the wiki with documentation and code samples!

11 Alternate Viewers

11.1 Preview.app

The Preview.app viewer is very straight-forward. It simply launches Preview.app with the relevant PDF file. Please note that Preview.app *does not* support forward or reverse sync, so you will not have that functionality available. Nevertheless, if you want to avoid installing another PDF viewer, this may be an acceptable option.

11.2 Okular

The Okular viewer is quite similar to the Evince viewer and should work out of the box. However, for forward sync (i.e. from Sublime to Okular) to work properly, the PDF document *must* be opened in Okular's unique session. If it is not, each forward sync command will open a new copy of the PDF. This also means that you can only have a single PDF document opened by LaTeXTools at a time. If, when the Okular viewer is run, you get a message which reads `There's already a unique Okular instance running. This instance won't be the unique one.`, you will need to adjust your `sync_wait` settings, increasing the value until the error stops. See the Linux platform settings.

11.3 Zathura

Zathura will mostly work out of the box. However, under some circumstances, Zathura may not properly gain focus if you have set `keep_focus` to `false` or set the toggle to `Focus PDF`. To ensure that the focus ends up on Zathura, you will have to install either `wmctrl` or `xodotool`, which should be available through your package manager. You can, of course, install both.

11.4 Command Viewer

Some support for other viewers is provided via the `command` viewer, which allows the execution of arbitrary commands to view a pdf or perform a forward search.

Using the command viewer requires that you configure the command(s) to be run in the platform-specific part of the `viewer_settings` block in your LaTeXTools preferences. There are three commands available:

- `forward_sync_command`: the command to executing a forward search (`ctrl + l, j` or `cmd + l, j`).
- `view_command`: the command to simply view the PDF document.

Of these, on `view_command` needs to be specified, though you will not have forward search capabilities unless you specify a `forward_sync_command` as well.

The following variables will be substitued with appropriate values inside your commands:

Variable	Description
`$pdf_file`	full path of PDF file, e.g. *C:\Files\document.pdf*
`$pdf_file_name`	name of the PDF file, e.g. *document.pdf*
`$pdf_file_ext`	extension of the PDF file, e.g. *pdf*
`$pdf_file_base_name`	name of the PDF file without the extension, e.g. *document*
`$pdf_file_path`	full path to directory containing PDF file, e.g. *C:\Files*
`$sublime_binary`	full path to the Sublime binary

In addition, the following variables are available for the `forward_sync_command` only:

Variable	Description
`$src_file`	full path of the tex file, e.g. *C:\Files\document.tex*
`$src_file_name`	name of the tex file, e.g., *document.tex*
`$src_file_ext`	extension of the tex file, e.g. *tex*
`$src_file_base_name`	name of the tex file without the extension, e.g. *document*
`$src_file_path`	full path to directory containing tex file, e.g. *C:\Files*
`$line`	line to sync to
`$col`	column to sync to

If none of these variables occur in the command string, the `$pdf_file` will be appended to the end of the command.

Commands are executed in the `$pdf_file_path`, i.e., the folder containing the `$pdf_file`.

For example, you can use the command viewer to support Okular with the following settings:

```
{
    "viewer": "command",

    "viewer_settings": {
        "linux": {
            "forward_sync_command": "okular --unique $pdf_file#src:$line$src_file",
            "view_command": "okular --unique"
        }
    }
}
```

12 LaTeXTools Cache

LaTeXTools uses a cache to store relevant information about your document and improve the performance of commands. The cache is made up of a series of files. Where these files are stored depends on your settings. By default, we try to keep them invisible, so they are stored in the Sublime cache path. However, by changing the `hide_local_cache` setting, you can have them stored in a hidden folder in the same directory as your tex root.

The local cache also has a lifespan, after which it will be invalidated. The lifespan starts when the first entry is inserted in the cache and the whole cache will be deleted after the lifespan. This can be set in the `local_cache_life_span` setting. The format is `"X d X h X m X s"`, where `X` is a natural number `s` stands for seconds, `m` for minutes, `h` for hours, and `d` for days. Missing fields will be treated as 0 and white-spaces are optional. Hence you can write `"1 h 30 m"` to refresh the cached data every one and a half hours. If you use `"infinite"` the cache will not be invalidated automatically. A lower lifespan will produce results, which are more up to date. However it requires more recalculations and might decrease the performance.

```
        self.display("done.\n")
```

13 Troubleshooting

13.1 Path issues

Many LaTeXTools problems are path-related. The `LaTeXTools.sublime-settings` file attempts to set up default path locations for MiKTeX, TeXLive and MacTeX, but these are not guaranteed to cover all possibilities. Please let me know if you have any difficulties.

On Mac OS X, just having your `$PATH` set up correctly in a shell (i.e., in Terminal) does *not* guarantee that things will work when you invoke commands from ST. If something seems to work when you invoke `pdflatex` or `latexmk` from the Terminal, but building from within ST fails, you most likely have a path configuration issue. One way to test this is to launch ST from the Terminal, typing

```
/Applications/Sublime Text.app/Contents/SharedSupport/bin/subl
```

(and then Return; this is for ST23of course) at the prompt. If things do work when you run ST this way, but they fail if you launch ST from the Dock or the Finder, then there is a path problem. From the Terminal, type

```
echo $PATH
```

and take note of what you get. Then, run ST from the Dock or Finder, open the console (with `Ctrl+ ``) and type

```
import os; os.environ['PATH']
```

and again take note of what you see in the output panel (right above the line where you typed the above command). Finally, look at the `texpath` keyword in the `osx` section of the `LaTeXTools.sublime-settings` file For things to work, every directory that you see listed from the Terminal must be either in the list displayed when you type the `import os...` command in the ST console, or else it must be explicitly specified in the `texpath` setting found in `LaTeXTools.sublime-settings`. If this is not the case, add the relevant paths to the `texpath` setting and *please let me know*, so I can decide whether to add the path specification to the default build file. Thanks!

On Linux, do note that your login shell may be different from the shell that launched Sublime Text. This can mean that LaTeXTools does not inherit your `$PATH` correctly, particularly if you modify your `$PATH` in `.bash_profile` or `.bashrc` or other, shell-specific files (X Windows is run via `/bin/sh` rather than `/bin/bash`). If you have a similar problem, follow the same procedure as above, although you should launch the `sublime_text` executable from a shell.

13.2 Non-ASCII characters and spaces in path and file names

Another *significant* source of issues are **Unicode characters in path and file names**. On TeXLive-based platforms, LaTeXTools tries to handle these by

telling `latexmk` to `cd` to each source file's directory before running `pdflatex`. This seems to help some. However, things seem to vary by platform and locale, so I cannot make any guarantees that your Unicode path names will work. Keep in mind that TeX itself has issues with Unicode characters in file names (as a quick Google search will confirm).

Spaces in paths and file names *are* supported. As far as I know, the only limitation has to do with multifile documents: the root document's file name cannot contain spaces, or the `%!TEX = <name>` directive will fail. I may fix this at some point, but for now it is a limitation.

13.3 Compilation hangs on Windows

On Windows, sometimes a build seems to succeed, but the PDF file is not updated. This is most often the case if there is a stale pdflatex process running; a symptom is the appearence of a file with extension `.synctex.gz(busy)`. If so, launch the Task Manager and end the `pdflatex.exe` process; if you see a `perl.exe` process, end that, too. This kind of behavior is probably a bug: LaTeXTools should be able to see that something went wrong in the earlier compilation. So, *please let me know*, and provide me with as much detail as you can (ideally, with a test case). Thanks!

13.4 Log parsing issues, and good vs. bad path/file names (again!)

As noted in the Highlights, the new parser is more robust and flexible than the old one—it "understands" the log file format much, much better. This is the result of *manually* and *painstakingly* debugging a fair number of users' log files. The many possible exceptions, idiosyncrasies, warts, etc. displayed by TeX packages is mind-boggling, and the parsing code reflects this :-(

Anyway, hopefully, errors should now occur only in strange edge cases. Please *let me know on github* if you see an error message. I need a log file to diagnose the problem; please upload it to gist, dropbox, or similar, and paste a link in your message on github. Issue #104 is open for that purpose.

There are *two exceptions* to this request. First, the *xypic* package is very, very badly behaved. I have spent more time debugging log files contaminated by xypic than I have spent fixing all other issues. Seriously. Therefore, first, parsing issues are now reported as "warnings" if the xypic package is used (so compilation and previewing continues); second, I cannot promise I will fix the issue even if you report it. Thanks for your understanding.

The second exception has to do with file and path names. In order to accommodate the many possible naming conventions across platforms and packages, as well as the different ways in which file names can occur in logs, I had to make some assumptions. The key one is that *extensions cannot contain spaces*. The reason is that the regex matching file names uses a period (".") followed by non-space characters, followed by a space as denoting the end of the file name. Trust me, it's the most robust regex I could come up with. So, you can have spaces in your base names, and you can even have multiple extensions; however,

you cannot have spaces in your extensions. So, “This is a file.ver-1.tex” is OK; “file.my ext” (where “my ext” is supposed to be the extension) is *not OK*.

Finally, I have done my best to accommodate non-ASCII characters in logs. I cannot promise that everything works, but I’d like to know if you see issues with this.