
TikZ Sphinx Extension Documentation

Release 0.4.6

Christoph Reller

Aug 05, 2018

Contents

1	Prerequisites and Configuration	3
1.1	Prerequisites	3
1.2	Ubuntu Linux	3
1.3	Mac OS X	4
1.4	Windows	4
1.5	Configuration	4
2	Usage	7
3	Examples	9
4	Caveats	11

This extension to [Sphinx](#) enables the use of the PGF/TikZ LaTeX package to draw nice pictures. (See [CTAN](#) or [sourceforge](#); the manual is, e.g., [here](#). Also have a look at contributions such as [pgfplots](#).)

Use the extension at your own risk. Anything might change in future versions without further notice.

Version 0.4.6

Author Christoph Reller christoph.reller@gmail.com

License [BSD License](#)

Git Repository <https://bitbucket.org/philexander/tikz>

PyPI Package <http://pypi.python.org/pypi/sphinxcontrib-tikz>

Documentation <http://sphinxcontrib-tikz.readthedocs.io>

Prerequisites and Configuration

1.1 Prerequisites

This extension relies on two software packages being installed on your computer:

1. `latex` with the `tikz` and the `amsmath` packages
2. A software package that is able to convert a PDF to an image. Currently, this extension supports four different ways of doing this conversion. We call them conversion “suites” and list for each suite what must be installed on your computer: (Only one such suite need be installed.)
 - pdf2svg suite: `pdf2svg`
 - Netpbm suite: `pdftoppm` (part of the Poppler pdf library) and `pnmtopng` (part of the Netpbm package)
 - ImageMagick suite: `pdftoppm` (part of the Poppler pdf library) and `convert` (part of the ImageMagick package)
 - GhostScript suite: `ghostscript`

1.2 Ubuntu Linux

For **Ubuntu Linux** you roughly have to make sure that the following packages are installed:

1. `texlive` and `texlive-pictures` (and maybe more LaTeX packages)
2. Depending on the chosen conversion suite the following package(s) have to be installed:
 - pdf2svg suite: `pdf2svg`
 - Netpbm suite: `poppler-utils` and `netpbm`
 - ImageMagick suite: `poppler-utils` and `imagemagick`
 - GhostScript suite: `ghostscript`

1.3 Mac OS X

For **Mac OS X** a possible way of getting this extension working is as follows:

1. Install the [MacTeX](#) LaTeX distribution which per default comes with the `tikz` package.
2. To install one of the conversion suites you can install [homebrew](#) and then use homebrew to install the package(s) listed under B. as above for Ubuntu Linux.

1.4 Windows

For **Windows** do the following:

1. Install the [MiKTeX](#) LaTeX distribution and include the `tikz` package when installing.
2. Depending on the chosen conversion suite, you have to install the following:

- pdf2svg suite:

Get the Windows binaries from [GitHub](#) copy all the files to some directory and add this directory to the `PATH` environment variable.

- Netpbm suite:

- [Xpdf package](#)
- [NetPbm for Windows package](#)

If you don't want to install the full packages above, you can copy the following files to some directory and add this directory to the `PATH` environment variable:

From Xpdf:

- `pdftoppm`

From NetPbm:

- `pnmtopng.exe`
- `libnetpbm10.dll`
- `libpng13.dll`
- `rgb.txt`

Also, you need to create a new environment variable `RGBDEF=C:\TikzSphinx\rgb.txt` assuming you copy the files to the `C:\TikzSphinx` directory.

- ImageMagick suite:

Install the [Xpdf package](#) (same as for the Netpbm suite) and install ImageMagick from [here](#).

- GhostScript suite:

Get the GhostScript binary from [here](#), copy it to some directory and add this directory to the `PATH` environment variable.

1.5 Configuration

If you have installed the Tikz Sphinx extension e.g. using [PyPI](#), then you have to load the extension in the Sphinx project configuration file `conf.py` by:


```
extensions = ['sphinxcontrib.tikz']
```

Additionally, the following configuration values are supported:

- Choose the image processing `<suite>`, either `'pdf2svg'`, `'Netpbm'`, `'ImageMagick'`, `'GhostScript'` (`'pdf2svg'` by default):

```
tikz_proc_suite = <suite>
```

Note:

- If you want your documentation to be built on <http://readthedocs.org>, you have to choose GhostScript.
- All suites produce png images, excepted `'pdf2svg'` which produces svg.

-
- Enable/disable transparent graphics (enabled by default):

```
tikz_transparent = <True or False>
```

- Add `<string>` to the LaTeX preamble used for building the TikZ picture:

```
tikz_latex_preamble = <string>
```

- Add `\usetikzlibrary{<string>}` to the LaTeX preamble used for building the TikZ picture:

```
tikz_tikzlibraries = <string>
```

Note: If you want to use the `latex` target, then you have to take care to include in `tikz_libraries` any `<tikz libraries>` given to the `libs` option of the `tikz` directive (see [Usage](#))

Note: If you want to make use of the TikZ externalization library for the LaTeX build output, then you may want to change the line:

```
LATEXOPTS =
```

in Sphinx LaTeX Makefile (`/usr/share/sphinx/texinputs/Makefile`) to:

```
LATEXOPTS = "-shell-escape"
```

The extension adds a `tikz`-directive and a `tikz`-role.

The **tikz-directive** can be used in two ways:

```
.. tikz:: <tikz code, potentially broken  
   across lines>  
   :libs: <tikz libraries>  
   :stringsubst:
```

or:

```
.. tikz:: <caption, potentially broken  
   across lines>  
   :libs: <tikz libraries>  
   :stringsubst:  
  
   <tikz code, potentially broken  
   across lines>
```

The `<caption>` is optional, but if present it is printed as a picture caption below the picture.

The `:libs:` option expects its argument `<tikz libraries>` to be a comma separated list of Tikz libraries to use. If you want to build the LaTeX target then make sure to add these libraries to the configuration value `tikz_tikzlibraries` in `conf.py`.

The `:stringsubst:` option enables the following string substitution in the `<tikz code>`: Before processing the `<tikz code>` the string `$wd` or `$(wd)` is replaced by the project root directory. This is convenient when referring to some source file in the LaTeX code.

The `<tikz code>` is code according to the TikZ LaTeX package. It behaves as if inside a `tikzpicture` environment. The presence of `\begin{tikzpicture}` and `\end{tikzpicture}` is optional.

Alternatively to providing the `<tikz code>`, the `:include:` option can be used to import the code from a file:

```
.. tikz::<caption, potentially broken  
   across lines>  
   :libs: <tikz libraries>  
   :include: <filename>  
   :stringsubst:
```

The **tikz-role** is used as follows:

```
:tikz:`<tikz code>`
```

The `<tikz code>` is code according to the Tikz LaTeX package. It behaves as if inside a `\tikz` macro.

CHAPTER 3

Examples

Note: These examples only render in a Sphinx project with a proper configuration of the Tikz Sphinx extension.

```
.. tikz:: [>=latex',dotted,thick] \draw[->] (0,0) -- (1,1) -- (1,0)
-- (2,0);
:libs: arrows
```



```
.. tikz:: An Example Directive with Caption

\draw[thick,rounded corners=8pt]
(0,0)--(0,2)--(1,3.25)--(2,2)--(2,0)--(0,2)--(2,2)--(0,0)--(2,0);
```

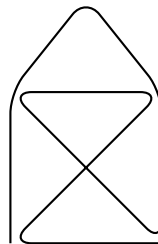
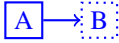
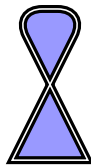


Fig. 1: An Example Directive with Caption

```
An example role :tikz:`[thick] \node[blue,draw] (a) {A};
\node[draw,dotted,right of=a] {B} edge[<-] (a);`
```

An example role 

Example of a Tikz picture included from a file:



CHAPTER 4

Caveats

If you use the `tikz` directive inside of a table or a sidebar and you specify a caption then the LaTeX target built by the sphinx builder will not compile. This is because, as soon as you specify a caption, the `tikzpicture` environment is set inside a `figure` environment and hence it is a float and cannot live inside a table or another float.

If you enable `:stringsubst:` and you happen to have any LaTeX math expression starting with `wd` (i.e., you would like to write `$wd \dots` then you must insert some white space, e.g., `$w d \dots` to prevent string substitution.