The packages **svg** and **svg-extract**

Philip Ilten (2012–2016)
Falk Hanisch (2017–)
https://github.com/mrpiggi/svg
hanisch.latex@outlook.com

v2.02k (2020/11/26)

The **svg** package is intended for the automated integration of SVG graphics into LaTeX documents. The capabilities provided by ***Inkscape***—or more precisely its command line interface—are used to export the text within a SVG graphic to a separate file, which is then rendered by LaTeX. The two commands `\includesvg` and `\includeinkscape` are provided as central user-interface, which are very similar to the `\includegraphics` command of the **graphicx** package.

In addition, the package **svg-extract** allows the extraction of these graphics into independent files in different graphic formats, exactly as it is rendered within the LaTeX document. For the creation of these graphics in the well-known formats PDF, EPS and PS, LaTeX and possibly conversion tools shipped with the distribution are used. If the graphics are required in other file formats, either ***ImageMagick*** or ***Ghostscript*** can be invoked.

The command line interface (CLI) of ***Inkscape*** 1.0 has changed in comparison to previous versions. In order to provide a comfortable user-interface for invoking ***Inkscape***, the used version is detected and if necessary switch to the outdated syntax of the CLI. If this approach fails for some reason, you can set the version of ***Inkscape*** manually with `inkscapeversion=0` or `inkscapeversion=1`.

Contents

Part I.
User documentation

1. Introduction

The open source program ***Inkscape*** has provided an excellent resource for the simple and easy creation of images and diagrams using a graphical user-interface. The work by Johan B. C. Engelen has further enhanced the ability of ***Inkscape*** to split a SVG file into a text component that can be compiled with LaTeX, and an image component that can be imported as a PDF file. For further information see the documentation of **svg-inkscape** on CTAN[1]. The procedure described therein is taken up and consistently expanded. Thus, it is now possible to include a SVG file into a LaTeX document where the text within the SVG graphic will be rendered natively by LaTeX.

---

[1] `http://www.ctan.org/pkg/svg-inkscape`

Both packages **svg** and **svg-extract** rely heavily upon executing commands on shell using the `\ShellEscape` command—or respectively the old known `\write18`—for executing the CLIs of the applications mentioned above. So passing flag `--shell-escape` to the LaTeX engine is utterly essential when using package **svg** and/or **svg-extract**. The executed commands and the possibilities to adapt their invocation with the appropriate options are described later on in this documentation. All this is done automatically with the `\includesvg` command. If you don't want to use the `--shell-escape` flag, either for security reasons or because the export of the SVG files is done in another way, there's also the command `\includeinkscape` which includes files already exported by ***Inkscape***.

An working installation of ***Inkscape*** is required for the automated integration of SVG graphics, whereby the installation path must be known to the operating system. This can be checked on shell by typing `inkscape -V`. Moreover, there are some required packages which are loaded by packages **svg** and **svg-extract** to provide the functionality. These are:

iftex for flow control depending on the used LaTeX engine
scrbase for the definition and handling of options in key-value-syntax
pdftexcmds, shellesc to allocate the same primitives independent of the used LaTeX engine
ifplatform to control the file access depending on the operating system
trimspaces to remove unwanted spaces in file paths
graphicx for including the graphic files after the ***Inkscape*** export
xcolor, transparent are possibly needed by the separate LaTeX files created by ***Inkscape***
xr is used by **svg-extract** in order to include labels within the independent graphic files

If you want to pass options to package **graphicx**, you must either load it before package **svg**

```
\usepackage[⟨options⟩]{graphicx}
...
\usepackage[⟨options⟩]{svg}
```

or use `\PassOptionsToPackage`.

```
\PassOptionsToPackage{⟨options⟩}{graphicx}
...
\documentclass[⟨options⟩]{⟨class⟩}
...
\usepackage[⟨options⟩]{svg}
```

The usage of packages **xcolor** and **transparent** can be switched off while loading package **svg**. See the two options `usexcolor` and `usetransparent` below.

2. Usage of package svg

The purpose of this package is to include standalone SVG graphics into a LaTeX document. The command `\includesvg` is defined which does all necessary steps for this task. It first launches the export of a SVG file to a supported file format with Inkscape, if necessary, and includes the exported graphic file afterwards. The usage and the syntax is quite similar to the command `\includegraphics` from the **graphicx** package. In fact, the inclusion of the exported graphic file is done with `\includegraphics`.

`usexcolor` (opt.)
`usetransparent` (opt.)
`noxcolor` (opt.)
`notransparent` (opt.)

The packages **xcolor** and **transparent** are loaded by default at the end of package **svg**. The listed options are intended to prevent these packages from loading. They are the only options which have to be given while loading the **svg** package. All supported boolean values (`true`/`on`/`yes`/`false`/`off`/`no`) can be assigned to `usexcolor` and `usetransparent`, while `noxcolor` and `notransparent` don't accept any value.

```
\usepackage[⟨options⟩]{svg}
```

Due to the way the LaTeX kernel parses package options, problems may occur if any option other than those just mentioned above – meaning the options explained hereafter – are passed through the optional argument of `\usepackage[`⟨*options*⟩`]{svg}`. It is strongly recommended to use `\svgsetup{`⟨*options*⟩`}` for these after loading the package instead.

2.1. General settings

`\svgsetup` All other options described in detail below can also be changed after loading the package either in the preamble or within the document. They don't have to be given as optional argument to `\usepackage[`⟨*options*⟩`]{svg}` but can be set by using macro `\svgsetup{`⟨*options*⟩`}` where ⟨*options*⟩ is a comma separated list of options. These settings are done in the current scope which means either globally or within the current group.

```
\svgsetup{⟨options⟩}
```

Further, with the optional argument of commands `\includesvg[`⟨*options*⟩`]{`⟨*svg filename*⟩`}` or `\includeinkscape[`⟨*options*⟩`]{`⟨*graphic filename*⟩`}`, it's possible to reset any setting locally for a certain file.

`\svgpath` Most likely you want to organize your SVG files in a separate folder either as a subfolder in the working directory or elsewhere in your local folder structure. For this purpose, a list of root paths to SVG files can be specified using the `\svgpath` command in the same way as `\graphicspath` is used. Every path has to be given in a group of braces `{}`—even if there is only one—and should terminate with a slash. For example:

```
\svgpath{{svg/}{/usr/local/svg/}}
```

would cause the system to look first in the subdirectory `svg/` and afterwards in the absolute path `/usr/local/svg/`. Further, if no path was specified with `\svgpath` or the desired file wasn't found, all directories given with `\graphicspath` are searched too. Please keep in mind that the current working directory is browsed first in any case. It is recommended to avoid umlauts or any other Non-ASCII characters as well as any spaces and/or quotes respectively `\dq` both in paths and file names. Especially when DVI output is active using quotes will certainly cause an error.

2.2. Options for the invocation of *Inkscape*

`inkscape` (opt.) This option controls, when the export with ***Inkscape*** is invoked and is *`true`* by default.

`false/off/no`
: ***Inkscape*** won't be invoked in any case, no export is done.

`true``/on/yes/newer/onlynewer`
: The export with ***Inkscape*** will only be done, if the exported graphic file either does not exist or the file modification date of the SVG file is newer than that of the exported graphic file. Thus the compilation time of the LaTeX document can be reduced to the necessary minimum.

`forced/force/overwrite`
: The ***Inkscape*** export will definitely be done, any already existing exported file will overwritten regardlessly.

In addition to controlling the export behavior, the option `inkscape` can also be used to make additional settings, which then acts as a wrapper for the options described below.

`pdf/eps/ps/png`
: see `inkscapeformat=pdf/eps/ps/png`

`latex/nolatex`
: see `inkscapelatex=true/false`

`drawing/page`
: see `inkscapearea=drawing/page`

⟨*`integer`*⟩`dpi`
: see `inkscapedpi=`⟨*`integer`*⟩

`inkscapepath` (opt.) The option `inkscapepath` specifies, where the resulting files of the ***Inkscape*** export should be located. The default setting is *`basesubdir`*, which uses the subfolder `./svg-inkscape/` within the current working directory.

`svgdir/svgpath`
: The PDF/EPS/PS/PNG graphic files as well as the LaTeX files generated by ***Inkscape*** will be located in the same directory as the corresponding SVG file.

`svgsubdir/svgsubpath`
: Within the folder of the encountered SVG file, all exported files will be located in a subfolder named `svg-inkscape/`.

`basedir/basepath/jobdir/jobpath`
: All exported files will be located in the current working directory.

`basesubdir``/basesubpath/jobsubdir/jobsubpath`
: A subfolder named `svg-inkscape/` within the current working directory will be used for files generated by ***Inkscape***.

`/path/to/somewhere/`
: It is also possible to give a custom path, either relative to the current working directory (`./relative/path/`) or as an absolute path.

`inkscapeexe` (opt.) For including a SVG file, ***Inkscape*** is used to separate the text and image from the SVG file itself. In order to use the command line interface on shell, the path where the executable is located has to be known to the operating system and is assumed to be `inkscape` by default.

You can check if the default setting is valid for your system by typing `inkscape -V` into the terminal. If this fails and nothing is returned, you should add the binary directory of ***Inkscape*** to the environment variable `PATH` on your operating system. For the case, that this is not possible or you aren't willing to do so, you can alternatively pass option `inkscapeexe` to `\svgsetup` *within the document preamble* to set the absolute path where the executable of ***Inkscape*** is located.

Especially if the executable path to be defined *contains spaces*, it *must not* be passed as a package option but to `\svgsetup{inkscapeexe=...}` instead!

`inkscapeversion` (opt.) The command line interface of ***Inkscape*** changed slightly from version `0.9x` to `1.x` and makes it necessary to distinguish between the two versions. By default, `inkscapeversion=auto` is set and the used version is automatically detected. This is done by calling ***Inkscape***-CLI with parameter `-V` on shell—see option `inkscapeexe` described above. The returned result is evaluated by either piping `stdout` or eventually—if this fails—writing to a temporary file and read this back in (pipes with a potentially quoted path can not be used with MiKTeX).[2] It is also possible to switch off the automatic detection routine by setting the desired version manually with either `inkscapeversion=0` to legacy mode or `inkscapeversion=1` to the current CLI version.

`inkscapename` (opt.) ***Inkscape*** export file names are derived from the SVG file name by default. However, the name of the exported file can be customized with `inkscapename=`⟨*`filename`*⟩. It is possible to use counters for specifying the name of the exported file. Repeatedly specifying the same file name will overwrite previously created files.

`inkscapeformat` (opt.) With this option, the ***Inkscape*** export format can be controlled. Valid values are `pdf`, `eps`, `ps` and `png`, where a LaTeX export is not possible for `png` and option `inkscapelatex` won't have any effect. By default, `inkscapeformat=pdf` is set unless DVI output was detected. In this case `inkscapeformat=eps` is the default setting.

`inkscapelatex` (opt.) If option `inkscapelatex=true` is set, the output is split into a separate PDF/EPS/PS file (see option `inkscapeformat`) and a corresponding LaTeX file. This is the default setting. Setting `inkscapelatex=false` will result in a single PDF/EPS/PS file, where any contained text won't be rendered by LaTeX.

`inkscapearea` (opt.) This option controls which area of the SVG file should be exported, *`drawing`* is set by default.

`drawing``/crop`
: The area exported corresponds to the bounding box of all objects in a drawing, including any that are not on the page.

`page/nocrop`
: The area exported will correspond to the defined page area within the SVG file.

---

[2] If this fails too, the ***Inkscape*** version is guessed when macro `\svg@ink@run` is used the very first time.

`inkscapedpi` (opt.)

The resolution used either for PNG export or for fallback rasterization of filtered objects when exporting to PDF/EPS/PS file. For PNG export it is set to 300 dpi by default, if no value was given. The given value should be a positive integer. The default behaviour can be reversed after a given value with `inkscapedpi=\relax`.

`inkscapeopt` (opt.)

You can use this option to pass additional switches to the ***Inkscape*** command line interface. For further information see the documentation of ***Inkscape***[3].

`svgextension` (opt.)

The package assumes SVG files with `.svg` extension as source for the ***Inkscape*** export. This option can be used to change this behaviour. For example, in order to process `.dia` files instead of `.svg` you could use

```
\includesvg[svgextension=dia,⟨additional options⟩]{⟨filename⟩}
```

2.3. Options for the graphic inclusion

`width` (opt.)
`height` (opt.)
`distort` (opt.)
`scale` (opt.)

The width of the included graphic file can be specified via the `width` option and the height by the `height` option. If both the width and height are specified, the figure will be scaled such that neither of the specified dimensions is exceeded, unless option `distort=true` is given.[4] If `width` and/or `height` once have been set, this can be undone by setting them to `0pt` or `\relax`. If neither `width` nor `height` are set, the included graphic file can also be scaled by setting `scale` to a positive real number.

`pretex` (opt.)
`apptex` (opt.)

Commands prior and post to the inclusion of the graphic file may be desired, such as font or color commands. The options `pretex` and `apptex` are provided where the LaTeX code given to `pretex` is included before the graphic file and `apptex` right afterwards. For example, to change the size of the included text one could use:

```
\includesvg[pretex=\tiny,⟨additional options⟩]{⟨svg filename⟩}
```

`draft` (opt.)

This option can be used with boolean values and is equal to the identically named option of the **graphicx** package. If the `draft` option is given to **graphicx**, it's activated for **svg** as well.

`lastpage` (opt.)

A bug[5] concerning the LaTeX export has been reported for ***Inkscape*** 0.91. It may happen that within the exported LaTeX file, it's attempted to include more pages of the PDF graphics than actually exist. The **svg** package attempts to bypass the resulting error.

Consequently, the total number of pages is read and only existing PDF pages are included, if both options `inkscapeformat=pdf` and `lastpage=true` are set. This is the default setting (unless DVI output is active) and can be switched off with `lastpage=false`. It's also possible to set the number of the last page included of a PDF graphic manually as optional parameter for `\includesvg` or `\includeinkscape`. For details, see the description of the respective commands.

2.4. Including SVG files

`\includesvg`

The command `\includesvg` to include a SVG file is quite similar to the `\includegraphics` command provided by the **graphicx** package.

```
\includesvg[⟨parameters⟩]{⟨svg filename⟩}
```

`inkscape` (param.)
`inkscapeformat` (param.)
`inkscapelatex` (param.)
`inkscapearea` (param.)
`inkscapedpi` (param.)
`inkscapeopt` (param.)
`svgextension` (param.)
`width` (param.)
`height` (param.)
`distort` (param.)
`scale` (param.)
`pretex` (param.)
`apptex` (param.)
`draft` (param.)

It is used right in the same way but where ⟨*svg filename*⟩ is the file name of the SVG file, where any given file extension will be replaced with `.svg` ruthlessly. In order to change the source file format for the ***Inkscape*** export, you have to use parameter `svgextension`.

If the given file is not located in the current working directory but elsewhere on your file system, the command `\svgpath` could be used to specify this path. It is recommended to avoid umlauts or any other Non-ASCII characters as well as any spaces and/or quotes

---

[3] `https://inkscape.org/de/doc/inkscape-man.html`

[4] to provide compatibility for package **graphicx**, it's possible to use `keepaspectratio=true` as alias for `distort=false` and the other way round

[5] `https://bugs.launchpad.net/ubuntu/+source/inkscape/+bug/1417470`

respectively `\dq` both in paths and file names. Especially when DVI output is active using quotes will certainly cause an error.

The command `\includesvg` is intended to do an automated export with ***Inkscape*** at first, where the given SVG file is exported to a PDF/EPS/PS/PNG file (see `inkscapeformat`) and perhaps a correlating LaTeX file (see `inkscapelatex`). The export with ***Inkscape*** is only invoked, if the SVG file is newer than the exported graphic file or latter doesn't exist at all. Once the export has been done, the graphic file and maybe the LaTeX file are included.

All previously described options can also be used as optional parameters to `\includesvg` and do have the same effect as described before. However, the optional parameters specified have an effect only once when `\includesvg` is executed and remain unchanged afterwards.

`lastpage` (param.)

In addition to the use of boolean values, the parameter `lastpage` can also be assigned a specific (integer) page number, which defines the last used page of a PDF graphic. This, just like the identically named option, has an effect only when `inkscapeformat=pdf` is set.

`angle` (param.)
`origin` (param.)

Both parameters correlate to the identically named parameters of the `\includegraphics` command provided by the **graphicx** package. However, unlike to `\includegraphics`, they `angle` and `origin` are *always evaluated after* `widht`, `height`, `distort` and `scale` by `\includesvg`, regardless of the used order of the given parameters. This is mainly due to the inclusion of the LaTeX files corresponding to the graphic files generated by ***Inkscape***.

2.5. Including already exported SVG files

`\includeinkscape`

If you don't want to make use of the automated export with ***Inkscape*** but the user-interface provided by the **svg** package, you can use `\includeinkscape` instead of `\includesvg`.

```
\includeinkscape[⟨parameters⟩]{⟨graphic filename⟩}
```

`inkscapeformat` (param.)
`inkscapelatex` (param.)
`width` (param.)
`height` (param.)
`distort` (param.)
`scale` (param.)
`pretex` (param.)
`apptex` (param.)
`draft` (param.)
`lastpage` (param.)
`angle` (param.)
`origin` (param.)

You can use it similar to `\includesvg` but ⟨*graphic filename*⟩ has to be the filename of the already exported graphic file. If a valid file extension (`.pdf`/`.eps`/`.ps`/`.png`) is given, the current setting for `inkscapeformat` is overwritten. It's even possible to specify a file extension like `.pdf_tex` to activate `inkscapelatex`. Furthermore, all optional parameters for `\includeinkscape` do have the same effect as described before for command `\includesvg` once when `\includeinkscape` is executed and remain unchanged afterwards.

3. Usage of package svg-extract

This package allows the extraction of independent graphic files out of SVG files which have been included and rendered with LaTeX by the **svg** package. This is particularly useful when attempting to provide images to journals or collaborators, and one wishes the image to appear exactly as it does within the original LaTeX document.

In order to extract to PDF, EPS, or PS files the programs `pstoeps`, `pstopdf` and `pdftops` are used which are usually provided by most of the LaTeX distributions. In addition, the command line interfaces of ***ImageMagick*** and ***Ghostscript*** can be invoked for converting images in formats like PNG, JPG, TIF or something else. It's also possible to create PDF, EPS or PS files with one of the two programs. Therefor the desired program—`magick` and/or `gswin32c`/`gswin64c` on Windows respectively `convert` and/or `gs` on unix-like operating systems—must be installed. By typing ⟨*program*⟩ `--version` on shell, this can be checked.

If you want to extract independent graphic files from included SVG files, you only have to load **svg-extract**. All actions for the extraction process will be done by using `\includesvg` or `\includeinkscape`. Without any additional settings, the extraction will render the SVG file to the specified output formats(s) of choice using the same settings as specified within the two commands. Consequently, the scale between the image and text in the extracted files will remain identical to the scale within the document from which the SVG file was extracted.

In contrast to package **svg**, the console commands for graphic extraction are executed with each LaTeX run by package **svg-extract** when `--shell-escape` mode is activated. This behaviour can be switched off with option `extract=false`.

Important changes

In version v1.0 of package **svg** the extracted files were named like the numbering of the current `subfig` environment by default. As package **subfig** sometime causes problems and because of the large amount of different LaTeX packages which all provide the possibility to include subfigures with very different implementations, this feature can't be provided reliably by **svg-extract**. See option `extractname` for further information.

3.1. General settings

`on` (opt.)
`off` (opt.)

This options have to be given while loading the **svg-extract** package and are intended to toggle the functionality of this package. As both extracting and converting independent graphic files is invoked with every LaTeX run when `--shell-escape` is activated, the option `off` can be given to save compilation time, once the creation of all desired images has been done and they no longer need to be re-generated. The option `on` can be used to reactivate functionality of this package. This can also be done by using `extract=true/false`.

`\svgsetup`
`\includesvg`
`\includeinkscape`

With package **svg-extract** the applicable options for `\svgsetup{`⟨*options*⟩`}` as well as parameters for the already described macros `\includesvg[`⟨*parameters*⟩`]{`⟨*svg filename*⟩`}` and `\includeinkscape[`⟨*parameters*⟩`]{`⟨*graphic filename*⟩`}` are extended. They can be used to control the process of graphic extraction and converting.

All options described below can be passed to `\svgsetup{`⟨*options*⟩`}` and are then valid in the current scope. There also exist identically named parameters for the optional arguments of

```
\includesvg[⟨parameters⟩]{⟨svg filename⟩}
\includeinkscape[⟨parameters⟩]{⟨graphic filename⟩}
```

These have an effect only once, when the specific command is executed.

3.2. Extract independent graphic files

`extract` (opt.)

This option can be used with boolean values. Using `extract=true` activates the functionality for both extracting and converting which is the default setting, whereas `extract=false` turns it off completely.

`extractpath` (opt.)

The path where the extracted and converted files are located can be specified with option `extractpath`, whereas *`basesubdir`* is set by default.

`svgdir/svgpath`
The extracted and converted independent graphic files are located in the same directory as the corresponding SVG file.

`svgsubdir/svgsubpath`
Within the folder of the encountered SVG file, all extracted and converted files will be located in a subfolder named `svg-extract/`.

`basedir/basepath/jobdir/jobpath`
All extracted and converted files will be located in the current working directory.

`basesubdir``/basesubpath/jobsubdir/jobsubpath`
A subfolder named `svg-extract/` within the current working directory will be used for all extracted and converted files.

`/path/to/somewhere/`
It is also possible to give a custom path, either relative to the current working directory (`./relative/path/`) or as an absolute path.

`extractname` (opt.)

It's also possible to change the name for extracted and converted files. The default setting is `extractname=filenamenumbered`. The appended file extension is derived from option `extractformat`.

`filename/name`
The name of the exported ***Inkscape*** file is used and the suffix `-extract` is attached.

`filenamenumbered`/`namenumbered`/`numberedfilename`/`numberedname`
: Same as above, but a prefix with the current enumerated count of SVG files is used instead of the suffix.

`numbered`/`section`/`numberedsection`/`sectionnumbered`
: The file name is composed by the current enumerated count of SVG files and the present outline numbering.

⟨*filename*⟩
: You can use any file name. It's possible to use counters for specifying the name of the extracted file. Repeatedly specifying the same file name will overwrite previously created files.

`extractformat` (opt.)

The included SVG file can be extracted from the document into an independent graphic file of type PDF, EPS or PS. The option can be used with either a single value (`extractformat=pdf`) or a comma separated list. For example,

```
\includesvg[extractformat={pdf,eps,ps}]{⟨svg filename⟩}
```

will extract the SVG file to both PDF and EPS formats and generates two independent graphic files. By default, `extractformat=pdf` is set unless DVI output was detected. In this case `extractformat=eps` is the default setting.

`extractwidth` (opt.)
`extractheight` (opt.)
`extractdistort` (opt.)
`extractscale` (opt.)
`extractpretex` (opt.)
`extractapptex` (opt.)

These options can be used to overwrite the settings given for the appearance of a SVG file within the document. For example, a SVG file should cover the entire text width within the document but be extracted to a fixed width. This can be done with:

```
\includesvg[width=\textwidth,extractwidth=500pt]{⟨svg filename⟩}
```

Assigning the value `inherit` to one of these options—which is set by default—leads to the usage of the corresponding option of package **svg** (`width`/`height`/`scale`/`pretex`/`apptex`), whereas `extract...=\relax` can be used to ignore a parent option utterly.

`extractpreamble` (opt.)
`extractpreambleend` (opt.)

Within the included and extracted SVG files any LaTeX macro can be used either defined by the user—this should be done in the preamble of the LaTeX document in which the SVG file is to be included—or provided by a package which is loaded. As the extraction process of the SVG files needs an auxiliary LaTeX file all used packages and commands have to be known within this file. Consequently, the preamble of the current LaTeX document is used for the extraction of the SVG file by default.

However, it is possible to specify a different *preamble file* with the option `extractpreamble` where the file to use as the preamble is given as the argument—including maybe path, but file name and file extension in any case. The given preamble file is searched similar to SVG files meaning, every path given with `\svgpath` or `\graphicspath` is examined. The default definition of `extractpreamble` is `\jobname.tex`—more precisely the file extension given by option `latexext` is used—and should suffice for most cases. The preamble up to the line defined by the option `extractpreambleend` will be used, which is set to a default with `\begin{document}`.

`\svghidepreamblestart`
`\svghidepreambleend`

In case, the preamble of the current LaTeX document is used, there are maybe packages included or some parts within the preamble, which should not be used within the separate auxiliary LaTeX file. These parts can be excluded if they are enclosed by `\svghidepreamblestart` and `\svghidepreambleend`.

For example, your current LaTeX document uses package **showframe** which causes some problems with the extraction of independent graphic files. So you want to get rid of it within the auxiliary LaTeX file. This can be done with:

```
\documentclass{⟨documentclassname⟩}
...
\usepackage{svg-extract}
...
\svghidepreamblestart
\usepackage{showframe}
\svghidepreambleend
...
```

`extractruns` (opt.) When extracting independent graphic files by compiling the generated auxiliary LaTeX file, it's maybe necessary to do multiple LaTeX iterations on this. The number of iterations is controlled with option `extractruns`. It is set to `extractruns=2` by default.

`latexexe` (opt.)
`latexopt` (opt.)
`latexext` (opt.)
For the extraction of an independent graphic file, the LaTeX program is used which is set by the `latexexe` option. Depending on the LaTeX engine used for the current LaTeX document, it is set to either ***pdflatex***, ***lualatex***, ***xelatex*** or ***latex*** by default. It's also possible to specify additional flags or switches for the LaTeX iterations, which are performed during the extraction process by the `latexopt` option. If you are used to utilize a different extension for LaTeX files than `.tex`, option `latexext` can be used like `latexext=ltx`.

`dvipsopt` (opt.)
`pstoepsopt` (opt.)
`pstopdfopt` (opt.)
`pdftoepsopt` (opt.)
`pdftopsopt` (opt.)
Depending on the used LaTeX engine, the file type of the extracted graphic differs. In order to create all formats, requested with option `extractformat`, several converting tools provided by most of the LaTeX distributions are maybe invoked. These are `dvips`, `ps2eps`, `ps2pdf` and/or `pdftops` and can't be changed. It's only possible to specify additional switches for every single tool with `dvipsopt`, `pstoepsopt`, `pstopdfopt`, `pdftoepsopt` and `pdftopsopt`.

`clean` (opt.) During the extraction process many files are generated for each SVG file extraction. So it's oftentimes desirable to automatically remove these temporary files. Using the option `clean=true` will remove any generated files created other than the extracted output format(s) requested. Setting `clean=false` is useful for debugging and set by default. Additionally, it's possible to use option `clean` with a list of file extensions in order to specify auxiliary files generated by package **svg-extract** to be deleted, for example `clean={log,aux}`.

`exclude` (opt.) Sometimes it may be necessary to extract and/or convert a SVG file without including it. If the flag `exclude` is specified, the SVG file will not be rendered in the current LaTeX document, but will be extracted and/or converted to the requested output format(s).

`\includesvg`
`\includeinkscape`
As previously mentioned, for extracting independent graphic files it is sufficient to load package **svg-extract** and afterwards everything necessary is done by just using `\includesvg` or `\includeinkscape`.

`extractangle` (param.) With this additional parameter the graphic is rotated during the extraction process. The value is not inherited from `angle` if it was given by default. This can be achieved by setting:

```
\includesvg[angle=⟨angle⟩,extractangle=inherit]{⟨svg filename⟩}
```

3.3. Convert extracted graphic files

Based on the extraction of independent graphic files, the **svg-extract** packages also provides the possibility to convert these extracted graphics in another format than PDF, EPS or PS with either ***ImageMagick***—which is set by default—or ***Ghostscript***.

`convert` (opt.) This option can be used to control the invocation of the conversion process. By default, `convert=false` is set. For Windows, there exist two different versions of ***Ghostscript***, either 64 bit or 32 bit. If it is selected as converting tool the 64 bit executable is set by default. Please note, that option `extract` has to be activated.

`false``/off/no`
: No conversion is done.

`true/on/yes`
: The conversion will be done with the current chosen converting tool.

`magick/imagemagick/convert`
: The conversion is activated and ***ImageMagick*** is selected.

`gs/ghostscript`
: The conversion is activated and ***Ghostscript*** is selected.

`gs64/ghostscript64`
: This value activates ***Ghostscript*** as conversion tool and sets `gsexe=gswin64c`. On unix-like operating systems, the value for `gsexe` remains unchanged.

`gs32/ghostscript32`
: The same as for the latter case applies, only option `gsexe=gswin32c` is set on Windows.

`convertformat` (opt.)

With this option, the desired output format(s) can be given. Multiple graphic formats can be specified in a list, for example something like `convertformat={png,jpg,tif}`. The value specified in `extractformat` is used as the source format for the conversion. If `extractformat` itself contains a file list, the first value within this list is considered. If `extractformat` is defined empty, the file generated anyway during the extraction is used.

Settings for specific converting formats

Maybe it's desired to apply varying settings for different output formats. Therefor some options described below can either be set for all converted files or for a specific output format. In particular, these are the options `convertdpi` as well as `magicksetting`, `magickoperator`, `gsdevice` and `gsopt`. All these mentioned options can be used like either `⟨option⟩=⟨value⟩` or `⟨option⟩={⟨outputformat⟩=⟨value⟩}` and even `⟨option⟩={⟨outputformat⟩+=⟨value⟩}` where the desired output format is trailed with `+` as inner key.

The first variant is applied to all output formats in general. If one of these mentioned options is evaluated and an output format specific value was given like in the second variant, the general setting is overwritten. If the general setting should be used and extended by an additional output format specific settings, then the third variant is to be used. In this case, no output format specific setting (second variant) must not have been used.

If you want to reverse any setting, you only have to use `\relax` as a value, either for a general option (`⟨option⟩=\relax`) or a specific one (`⟨option⟩={⟨outputformat⟩[+]=\relax}`).

`convertdpi` (opt.)

This option controls the used density for all file formats or a specific one, whether ***ImageMagick*** or ***Ghostscript*** is used for the graphic conversion. The desired resolution of the converted file is given in dots per inch (DPI) either as a scalar value (e.g. `convertdpi=600`) or with different resolutions in x- and y-direction (e.g. `convertdpi=600x400`).

As described before, it's also possible to declare a specific resolution for each desired converting format. For example, you want to set different resolution for PNG and JPG formats and something for all other formats:

```
\svgsetup{%
  convertdpi={png=600},%
  convertdpi={jpg=150},%
  convertdpi=300%
}%
```

If a setting for a specific output format is given, any unspecific setting is overwritten, when the conversion to this format is done. With `convertdpi={⟨outputformat⟩=\relax}` a specific setting can be reversed.

Please note that not every graphic format support different resolutions in x- and y-direction. So using a value like `convertdpi=600x400` may not necessarily lead to the desired result. However, this is then due to the used conversion tool and not to the processing of the option.

3.3.1. Settings for the invocation of *ImageMagick*

`magickexe` (opt.)
`magicksetting` (opt.)
`magickoperator` (opt.)

The conversion with ***ImageMagick*** via the `magick` or `convert` command line interface can be controlled with these options. The option `magickexe` determines the used executable and is set to `magick` on Windows and otherwise to `convert` by default. Additionally, there are the two options `magicksetting` and `magickoperator` which can be used to define *settings* and *operators* for the conversion process. As described before, the two options `magicksetting` and `magickoperator` can be set for all output formats or a *specific* one either resetting or extending the general settings. For further information see the documentation of ***ImageMagick*** command line interface[6].

---

[6] http://www.imagemagick.org/script/command-line-processing.php

3.3.2. Settings for the invocation of *Ghostscript*

`gsexe` (opt.)
`gsdevice` (opt.)
`gsopt` (opt.)

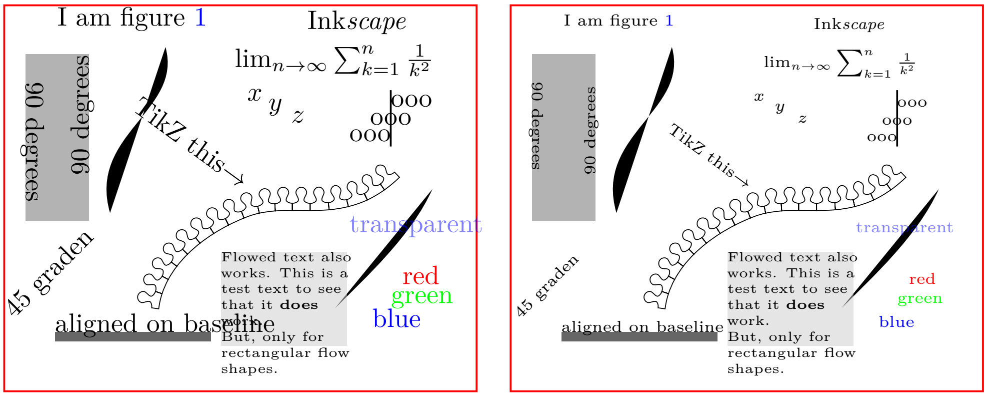
The conversion with ***Ghostscript*** is done with command line interface `gswin64c` or `gswin32c` on Windows and `gs` on unix-like operating systems. The executable can be changed with option `gsexe`. Because ***Ghostscript*** requires the specification of a device, there are some predefined for the most common output formats. These are:

```
\svgsetup{%
  gsdevice={png=png16m},gsdevice={jpeg=jpeg},gsdevice={jpg=jpeg},%
  gsdevice={tif=tiff48nc},gsdevice={tiff=tiff48nc},%
  gsdevice={eps=eps2write},gsdevice={ps=ps2write}%
}%
```

Furthermore, with `gsopt` additional switches for ***Ghostscript*** can be set. As described before, both `gsdevice` and `gsopt` can be defined in general or for specific output formats. For further information see the documentation of ***Ghostscript***[7].

---

[7] `https://ghostscript.com/doc/current/Use.htm`


I am figure 1
Inkscape
$\lim_{n\to\infty}\sum_{k=1}^{n}\frac{1}{k^2}$
x y z
ooo
ooo
ooo
90 degrees
90 degrees
TikZ this→
transparent
45 graden
Flowed text also works. This is a test text to see that it does work. But, only for rectangular flow shapes.
red
green
blue
aligned on baseline
I am figure 1
Inkscape
$\lim_{n\to\infty}\sum_{k=1}^{n}\frac{1}{k^2}$
x y z
ooo
ooo
ooo
90 degrees
90 degrees
TikZ this→
transparent
45 graden
Flowed text also works. This is a test text to see that it does work. But, only for rectangular flow shapes.
red
green
blue
aligned on baseline

(a) This text is too large!

(b) This text fits better.

Figure 1: An example figure with LaTeX support

4. Example

As an minimal example[8] take the following lines of code:

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage{svg}
\usepackage[off]{svg-extract}
\svgsetup{clean=true}
%\pdfsuppresswarningpagegroup=1
\usepackage{relsize}
\usepackage{subcaption}
\begin{document}
\begin{figure}
  \begin{minipage}{\dimexpr\linewidth/2\relax}
    \includesvg[width=\linewidth]{svg-example}%
    \subcaption{This text is too large!}%
  \end{minipage}%
  \begin{minipage}{\dimexpr\linewidth/2\relax}
    \includesvg[width=\linewidth,pretex=\relscale{0.6}]{svg-example}%
    \subcaption{This text fits better.}%
  \end{minipage}
\caption{An example figure with \LaTeX~support\label{fig:example}}%
\end{figure}
\begin{figure}\centering
  \includesvg[%
    width=.5\linewidth,inkscapelatex=false,extractformat={pdf,eps}%
  ]{svg-example}%
  \caption{The same example figure without \LaTeX~support}%
\end{figure}
\end{document}
```

The output is shown in Figure 1 and Figure 2. Within this example the file `svg-example.svg` was included three times using the `\includesvg` command.

If you are willing to compile the example, there are two aspects to consider. First, the included SVG file `svg-example.svg` has to be located in the current folder and is located in ⟨*texmf*⟩`/doc/latex/svg/`. Second, you have to run the desired LaTeX engine with flag `--shell-escape`.

As you can see, Figure 1a is created with default settings, except the width specification. The ***Inkscape*** export with LaTeX support is done and the extraction of an independent graphic file in PDF format as the **svg-extract** package was loaded.

---

[8] The image used here is a slightly modified version of the image used in the initial documentation on how to include a SVG file in LaTeX by Johan B. C. Engelen available as package **svg-inkscape** on CTAN.

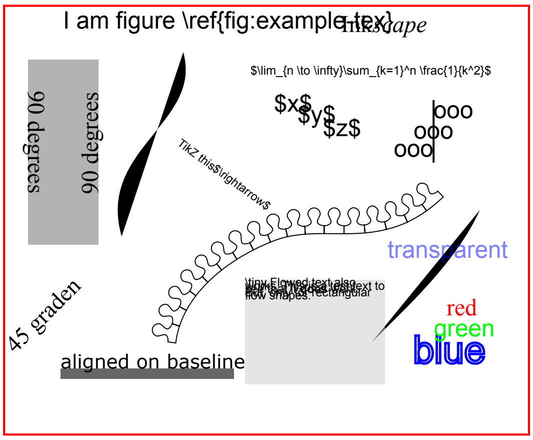


Figure 2: The same example figure without LATEX support

However, the text is slightly overrunning the margins of the image, and so Figure 1b—which again uses the same ***Inkscape*** export results—decreases the font size of the text within the image relative using the `pretex` option together with the `\relscale` command provided by the **relsize** package.

In Figure 2 the same SVG file was used but without the export of a separate LATEX file containing all text elements.

Feel free to use this given example to try out all the options and possibilities described in section 2 for package **svg**. Especially if you want to use package **svg-extract** for the automated extraction of independent graphics (subsection 3.2) and their conversion to different graphic formats with ***ImageMagick*** and/or ***Ghostscript*** (subsection 3.3), this example can be easily used for the first steps.

5. Troubleshooting and reporting issues

When using the packages **svg** and **svg-extract**, the most likely occurring problems will be caused by calling the external programs. For this reason, a short package information is written into the log file right before each call of an external program on shell. If a file should have been created, both packages check after the external call, whether this file exists or not and raise an error or at least a warning, if this file is missing. If you got such a message, please check the log file for lines like:

`Package svg Info:` or `Package svg-extract Info:`

Right afterwards, there should appear `runsystem(<command>)...excuted.` which you should try to execute manually at the terminal in the right directory. In most cases, the problem will be an invalid command call. If something goes wrong during the extraction/-converting process of package **svg-extract**, it would make sense to set option `clean=false` to not delete any auxiliary files that might be needed.

If you are sure that the problem is not caused by the configuration of your operating system, you can send an error report either via email or create a new issue on GitHub. Both addresses can be found on the title.

When using pdfLATEX there are a lot of warnings

It may happen that several warnings like

```
pdfTeX warning: pdflatex.exe(file ⟨filename⟩.pdf): PDF inclusion:
multiple pdfs with page group included in a single page
```

occur when including the PDF graphics exported with ***Inkscape***. This is related to the handling of transparency effects within PDF files. Since pdfTEX version 1.40.15 or later, you can get rid of these messages by using `\pdfsuppresswarningpagegroup=1`. See also the discussion on LaTeX Stack Exchange[9] for more information.

6. Include SVG files created with *ROOT*

This section was originally written by Philip Ilten. In the hope that since then nothing has changed fundamentally in the described procedure. This passage remains in the documentation, even if it will almost certainly be relevant to experimental particle physicists only, who frequently use the analysis package ***ROOT***.

ROOT has the ability to export directly to a SVG file, which means that it is possible to completely by-pass all of ***ROOT***'s internal text rendering machinery, and let LaTeX handle the text natively. This means that all of the ugly fonts that are rendered by ***ROOT*** can now be completely avoided, with the additional bonus of being able to add references within plots. So how does one go about using this package with ***ROOT***?

1. Create the plot with ***ROOT*** as normal, but turn off all LaTeX interpretation of text strings. This is a bit tricky, but can be accomplished by setting the font in ***ROOT*** to a precision of zero as described in the documentation for `TAttFill`[10]. Remember that the font is set by using the function `(TAttFill*)->SetTextFont(i)` with

   $$\texttt{i} = (\text{font type}) \times 10 + (\text{font precision})$$

   In the following lines of code, a `TStyle` is defined which sets the font to type "Courier New" with a precision of zero.

   ```
   TStyle *style = new TStyle("style","style"); int FONT = 80;
   style->SetTextFont(FONT);
   style->SetLabelFont(FONT,"XYZ");
   style->SetTitleFont(FONT,"XYZ");
   style->SetTitleFont(FONT,"");
   gROOT->SetStyle("style");
   gROOT->ForceStyle();
   ```

   Now, you can just use the well-known standard LaTeX syntax for creating labels, etc. Note however, that backslashes have to be escaped due to interpretation of special characters by ***C++***.

2. Print the plot as a SVG file.

   ```
   gPad->Print("foo.svg");
   ```

3. Include the SVG file within the document using this package.

   ```
   \usepackage{svg}
   \usepackage{svg-extract}
   \svgsetup{clean=true}
   ...
   \includesvg[width=\linewidth]{foo}
   ```

Consider the following example image produced by ***ROOT*** in Figure 3. This figure was generated by the ***ROOT*** macro `root.C`, provided within ⟨*texmf*⟩`/doc/latex/svg/`, which produces the file `root.svg` when run. The code used to produce this SVG file from within ***ROOT*** is

```
void root() {

  // Set the style.
  gStyle->SetTextFont(80);     gStyle->SetLabelFont(80,"XYZ");
  gStyle->SetTitleFont(80,""); gStyle->SetTitleFont(80,"XYZ");
  gStyle->SetPalette(1);       gStyle->SetOptStat(0);
```

---

[9] http://tex.stackexchange.com/questions/76273/

[10] http://root.cern.ch/root/html/TAttText.html

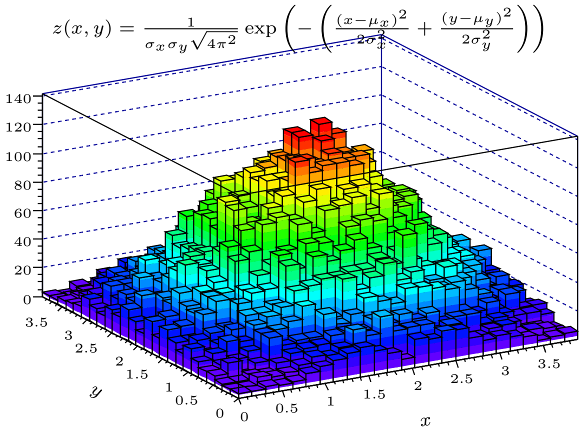


Figure 3: Rendering of a ***ROOT*** plot—no more *Comic CERNs*

```
  // Draw the plot.
  TH2D *h = new TH2D("", "", 25, 0, 3.9, 25, 0, 3.9); TRandom r;
  for (int i = 0; i < 30000; i++) h->Fill(r.Gaus(2.,1), r.Gaus(2.,1));
  h->GetXaxis()->CenterTitle(); h->GetXaxis()->SetTitleOffset(2.5);
  h->GetYaxis()->CenterTitle(); h->GetYaxis()->SetTitleOffset(2.5);
  h->GetXaxis()->SetTitle("\\larger[2]$x$");
  h->GetYaxis()->SetTitle("\\larger[2]$y$");
  h->Draw("LEGO2");

  // Draw additional text.
  TText *t = new TText(); t->SetTextAlign(31);
  t->DrawText(0.7, 0.9, "\\larger[2]$z(x,y) = \\frac{1}{\\sigma_x\\sigma_y"
                "\\sqrt{4\\pi^2}}\\exp\\left(- \\left(\\frac{(x-\\mu_x)^2}"
                "{2\\sigma_x^2} + \\frac{(y-\\mu_y)^2}{2\\sigma_y^2} \\right)"
                "\\right)$");

  // Print the plot.
  gPad->Print("root.svg");

}
```

where the text produced within the ***ROOT*** plot is set to a precision of zero.

The plot was then included within this document using the following LaTeX code

```
  \begin{figure}
    \centering%
    \includesvg[%
      inkscapearea=page,height=6cm,pretex=\tiny,convertformat=png%
    ]{root}%
    \caption{%
      Rendering of a \app{ROOT} plot---no more \emph{Comic CERNs}%
      \label{fig:root}%
    }%
  \end{figure}
```

which includes the graphic as well as the LaTeX file exported by ***Inkscape***, produces the extracted PDF image (`root.pdf`) and converts this to a PNG image (`root.png`) by using ***ImageMagick***. Enjoy plots from ***ROOT*** with natively rendered LaTeX!

Part II. Implementation

A. Initialization

A.1. Packages

The package **svg** mainly requires **scrbase** for options processing and **graphicx** for the inclusion of the exported graphic files.

The packages **iftex** and **pdftexcmds** are needed to detect the used LaTeX engine on the one hand and enabling pdfTeX primitives independent of the used LaTeX engine on the other hand. Additionally, **trimspaces** is responsible for string manipulation. Both packages **shellesc** and **ifplatform** are used for engine independent access to systems commands and files. The package **svg-extract** only needs package **svg** itself, which is loaded during initialization.

```
⟨*main⟩
\RequirePackage{iftex}[2020/03/06]
\RequirePackage{scrbase}[2020/09/21]
\RequirePackage{pdftexcmds}[2019/11/24]
\RequirePackage{trimspaces}[2009/09/17]
\RequirePackage{graphicx}[2019/11/30]
\RequirePackage{shellesc}[2019/11/08]
```

In order to do not raise a warning, package **ifplatform** is only used if `--shell-escape` flag is enabled.

```
\ifnum\pdf@shellescape=\@ne\relax
  \RequirePackage{ifplatform}[2017/10/13]
\fi
⟨/main⟩
```

A.2. Dealing with catcodes

The catcode for double quotes are temporarily changed and restored at the very end of both packages.

```
\edef\svg@catcodecodes@restore{%
  \catcode‘\noexpand\"\the\catcode‘\"\relax%
}
\@makeother\"%
\AtEndOfPackage{\svg@catcodecodes@restore}
```

A.3. General macros

`\svg@tempa`
`\svg@tempb`
`\if@svg@tempswa`

Internal temporary macros.

```
\newcommand*\svg@tempa{}
\newcommand*\svg@tempb{}
\newif\if@svg@tempswa
```

A.3.1. Macros for process control

`\svg@ifwindowsdetected` Do some Windows specific stuff if it was detected.

```
\newcommand*\svg@ifwindowsdetected{\@secondoftwo}
\AfterPackage*{ifplatform}{%
  \renewcommand*\svg@ifwindowsdetected{%
    \ifwindows%
      \expandafter\@firstoftwo%
    \else%
      \expandafter\@secondoftwo%
    \fi%
  }%
}
```

`\svg@ifvalueisrelax` For some keys the usage of `\relax` as a value should lead to a special reaction, such as restoring to default behavior or resetting the key. Therefore, `\svg@ifvalueisrelax` checks, whether `\relax` was used as value or not.

```
\newcommand*\svg@ifvalueisrelax[1]{%
  \begingroup%
    \def\svg@tempa{#1}%
    \def\svg@tempb{\relax}%
    \ifx\svg@tempa\svg@tempb%
      \aftergroup\@firstoftwo%
    \else%
      \aftergroup\@secondoftwo%
    \fi%
  \endgroup%
}
```

`\svgx@ifkeyandval`
`\svgx@@ifkeyandval` It is checked whether a key was given as ⟨*key*⟩=⟨*value*⟩ or like ⟨*key*⟩={⟨*format*⟩=⟨*value*⟩}.

```
\newcommand*\svgx@@ifkeyandval{}
\newcommand*\svgx@ifkeyandval[3]{%
  \def\svgx@@ifkeyandval##1=##2=##3\@nil{\IfArgIsEmpty{##3}{#3}{#2}}%
  \svgx@@ifkeyandval#1==\@nil%
}
```

A.3.2. String manipulation

Both packages **svg** and **svg-extract** should be able to handle user-defined input and output paths. As there is the possibility for users to provide paths with or without quotes to LaTeX, this is taken into account.

`\svg@deactivate@dq` In order to avoid errors concerning file names with package **babel** and its active double quotes, this command is defined.

```
\newcommand*\svg@deactivate@dq{}
\AfterAtEndOfPackage*{babel}{%
  \renewcommand*\svg@deactivate@dq{\bbl@deactivate{"}}%
  \providecommand*\bbl@deactivate[1]{}%
}
```

`\svg@sanitize@dq` Save expansion of the second argument in the macro from the first argument with deactivated double quotes.

```
\newcommand*\svg@sanitize@dq[2]{%
  \begingroup%
    \svg@deactivate@dq%
    \edef\svg@tempa{\endgroup\def\noexpand#1{#2}}%
  \svg@tempa%
}
```

`\svg@quotes@check`
`\svg@quotes@@check`
`\if@svg@quotes@found`

During the treatment of paths, it may be necessary to temporarily remove quotes and, if required, add them again later. For this purpose, the switch `\if@svg@quotes@found` as well as the commands `\svg@quotes@check` and `\svg@quotes@@check`, which controls the switch, are defined. As before, the string is passed in a macro to `\svg@quotes@check`.

```
\newif\if@svg@quotes@found
\newcommand*\svg@quotes@check[1]{%
  \expandafter\svg@quotes@@check#1"\@nil%
}
\newcommand*\svg@quotes@@check{}
\def\svg@quotes@@check#1"#2\@nil{%
  \IfArgIsEmpty{#2}{\@svg@quotes@foundfalse}{\@svg@quotes@foundtrue}%
}
```

`\svg@quotes@remove`
`\svg@quotes@@remove`

These two commands are used to remove all occurring quotes within a string. The only argument passed to `\svg@quotes@remove` is not the string itself but a macro in which a string is stored.

```
\newcommand*\svg@quotes@remove[2][]{%
  \begingroup%
    \IfArgIsEmpty{#1}{\def\svg@tempb{#2}}{\def\svg@tempb{#1}}%
    \svg@sanitize@dq\svg@tempa{\svg@tempb}%
    \expandafter\svg@quotes@check\expandafter{\svg@tempa}%
    \expandafter\svg@quotes@@remove\svg@tempa""\@nil%
    \edef\svg@tempb{%
      \endgroup%
      \def\noexpand#2{\svg@tempa}%
      \if@svg@quotes@found%
        \noexpand\@svg@quotes@foundtrue%
      \else%
        \noexpand\@svg@quotes@foundfalse%
      \fi%
    }%
  \svg@tempb%
}
\newcommand*\svg@quotes@@remove{}
\def\svg@quotes@@remove#1"#2"#3\@nil{%
  \IfArgIsEmpty{#2}{%
    \edef\svg@tempa{#1}%
  }{%
    \svg@quotes@@remove#1#2#3""\@nil%
  }%
}
```

`\svg@remove@leadingchar`

This command removes the single character in given with the first argument from the expanded macro in the second argument.

```
\newcommand*\svg@remove@leadingchar[2]{%
  \begingroup%
    \svg@sanitize@dq\svg@tempa{#2}%
    \def\svg@tempb{%
      \def\svg@tempa####1\@nil{\def\svg@tempa{####1}}%
      \kernel@ifnextchar#1%
        {\expandafter\svg@tempa\@gobble}%
        {\svg@tempa}%
    }%
    \expandafter\svg@tempb\svg@tempa\@nil%
    \edef\svg@tempb{%
      \endgroup%
      \def\noexpand#2{\svg@tempa}%
    }%
  \svg@tempb%
}
```

A.3.3. File handling

`\svg@filename@parse`

As the internal LaTeX command `\filename@parse` is not able to split a given file name containing quotes, `\svg@filename@parse` is defined to resolve this problem. The optional argument can be used to give a specific file extension, which should be searched within `\filename@ext`. If found at the very end, the previous part is appended to `\filename@base`.

```
\newcommand*\svg@filename@parse[2][]{%
  \begingroup%
```

The given path and file is parsed with `\filename@parse`.

```
    \svg@sanitize@dq\svg@tempa{#2}%
    \expandafter\filename@parse\expandafter{\svg@tempa}%
% If there are quotes in the file path, the closing one will be found as first
% character in \cs{filename@base} as \cs{filename@area} is split at the last
% slash. This leading quote is removed from \cs{filename@base} with
% \cs{svg@remove@leadingchar}.
%    \begin{macrocode}
    \svg@quotes@remove{\filename@area}%
    \if@svg@quotes@found%
      \edef\filename@area{"\filename@area"}%
      \svg@remove@leadingchar"\filename@base%
    \fi%
```

The found extension is parsed against the optional argument. If a double quote was found within the extension, it actually belongs to `\filename@base`.

```
    \ifx\filename@ext\relax\else%
      \svg@quotes@remove{\filename@ext}%
      \svg@extension@parse{#1}%
      \if@svg@quotes@found%
        \edef\filename@base{\filename@base"}%
      \fi%
    \fi%
```

Quotes within `\filename@base` are normalized.

```
    \svg@quotes@remove{\filename@base}%
    \if@svg@quotes@found%
      \edef\filename@base{"\filename@base"}%
    \fi%
```

With `\svg@tempa` the group is closed and the results are saved in the macros `\filename@...`.

```
    \edef\svg@tempa{%
      \endgroup%
      \def\noexpand\filename@area{\filename@area}%
      \def\noexpand\filename@base{\filename@base}%
      \ifx\filename@ext\relax%
        \let\noexpand\filename@ext\noexpand\relax%
      \else%
        \def\noexpand\filename@ext{\filename@ext}%
      \fi%
    }%
  \svg@tempa%
}
```

`\svg@extension@parse`
`\svg@extension@@parse`

These macros are used to permit multiple dots in file names. The content of `\filename@ext` is split at each occurrence of `.` and the trailing part is compared against the content of the argument of `\svg@extension@parse`, which is probably `\svg@file@ext`. If they are equal, the previous part is appended to `\filename@base` and `\filename@ext` is set to the content of the first argument.

```
\newcommand*\svg@extension@parse[1]{%
  \IfArgIsEmpty{#1}{}{%
```

```
    \@expandtwoargs\Ifstr%
      {\detokenize\expandafter{\filename@ext}}{\detokenize\expandafter{#1}}{}{%
      \begingroup%
```

Macro `\svg@tempa` is used to temporarily store anything before the searched extension at the end of `\filename@ext` and `\svg@tempb` is set to the actual searched extension if found.

```
        \edef\svg@tempa{%
          \def\noexpand\svg@tempa{}%
          \let\noexpand\svg@tempb\relax%
          \noexpand\svg@extension@@parse%
            \filename@ext.\noexpand\@nil#1\noexpand\@nil%
        }%
        \svg@tempa%
        \edef\svg@tempa{%
          \endgroup%
```

If the trailing extension was found, `\filename@base` and `\filename@ext` are adopted.

```
          \def\noexpand\filename@base{\filename@base\svg@tempa}%
          \ifx\svg@tempb\relax%
            \let\noexpand\filename@ext\relax%
          \else%
            \def\noexpand\filename@ext{\svg@tempb}%
          \fi%
        }%
      \svg@tempa%
    }%
  }%
}
```

Macro `\svg@extension@@parse` is recursively called as long as there are any dots or the searched extension is found.

```
\newcommand*\svg@extension@@parse{}
\def\svg@extension@@parse#1.#2\@nil#3\@nil{%
  \edef\svg@tempa{\svg@tempa.#1}%
  \IfArgIsEmpty{#2}{}{%
    \Ifstr{\detokenize{#2}}{\detokenize{#3.}}{%
```

If the trailing extension is found, `\svg@tempb` is defined.

```
      \edef\svg@tempb{#3}%
    }{%
      \svg@extension@@parse#2\@nil#3\@nil%
    }%
  }%
}
```

`\svg@iffilenewer`

The macro `\svg@iffilenewer` is used to decide, whether the export with ***Inkscape*** is necessary due to an updated SVG file. This can only be done, if `\pdf@filemoddate` or `\filemoddate` is defined.

```
\newcommand*\svg@iffilenewer[2]{\@gobbletwo}
\ifx\pdf@filemoddate\@undefined
  \ifx\filemoddate\@undefined\else
    \ifx\strcmp\@undefined\else
      \renewcommand*\svg@iffilenewer[2]{%
        \begingroup%
          \edef\svg@tempa{\filemoddate{#1}}%
          \edef\svg@tempb{\filemoddate{#2}}%
          \ifnum\strcmp{\svg@tempa}{\svg@tempb}>\z@\relax%
            \aftergroup\@firstoftwo%
          \else%
            \aftergroup\@secondoftwo%
          \fi%
        \endgroup%
      }%
    \fi
  \fi
\else
  \ifx\pdf@strcmp\@undefined\else
    \renewcommand*\svg@iffilenewer[2]{%
      \begingroup%
        \edef\svg@tempa{\pdf@filemoddate{#1}}%
        \edef\svg@tempb{\pdf@filemoddate{#2}}%
        \ifnum\pdf@strcmp{\svg@tempa}{\svg@tempb}>\z@\relax%
          \aftergroup\@firstoftwo%
        \else%
          \aftergroup\@secondoftwo%
        \fi%
      \endgroup%
    }%
  \fi
\fi
```

`\svg@shell@mkdir`
`\svg@shell@@mkdir`
`\svg@shell@mv`
`\svg@shell@@mv`
`\svg@shell@rm`
`\svg@shell@@rm`

Finally, platform dependent macros for creating directories as well as moving and deleting files are provided.

```
\newcommand*\svg@shell@mkdir[1]{%
  \begingroup%
```

A directory should only be created, if it isn't the current working directory.

```
    \svg@quotes@remove[{#1}]{\svg@tempa}%
    \@svg@tempswatrue%
    \Ifstr{\svg@tempa}{}{\@svg@tempswafalse}{%
    \Ifstr{\svg@tempa}{./}{\@svg@tempswafalse}{%
    }}%
    \if@svg@tempswa%
      \ShellEscape{\svg@shell@@mkdir{\svg@tempa}}%
    \fi%
  \endgroup%
}
\newcommand*\svg@shell@mv[2]{%
  \ShellEscape{\svg@shell@@mv\space"#1"\space"#2"}%
}
\newcommand*\svg@shell@rm[1]{%
  \ShellEscape{\svg@shell@@rm\space"#1"}%
}
```

The platform dependent commands for file access.

```
\svg@ifwindowsdetected{%
  \newcommand*\svg@shell@@mkdir[1]{if not exist "#1" mkdir "#1"}%
  \newcommand*\svg@shell@@mv{move}%
  \newcommand*\svg@shell@@rm{del}%
}{%
  \newcommand*\svg@shell@@mkdir[1]{mkdir -p "#1"}%
  \newcommand*\svg@shell@@mv{mv}%
  \newcommand*\svg@shell@@rm{rm}%
}
```

`\svg@normalize@path`
`\svg@normalize@@path`

If any path is given, a trailing slash is needed. These two macros ensure that this condition is fulfilled in any case, even if this is not considered by the user. As before, a macro containing the path string is passed to `\svg@normalize@path`.

```
\newcommand*\svg@normalize@path[1]{%
  \begingroup%
    \svg@quotes@remove[{#1}]{\svg@tempa}%
    \ifx\svg@tempa\@empty\relax%
```

```
      \def\svg@tempa{./}%
    \fi%
    \expandafter\svg@normalize@@path\svg@tempa//\@nil%
    \edef\svg@tempb{%
      \endgroup%
      \if@svg@quotes@found%
        \def\noexpand#1{"\svg@tempa"}%
      \else%
        \def\noexpand#1{\svg@tempa}%
      \fi%
    }%
  \svg@tempb%
}
\newcommand*\svg@normalize@@path{}
\def\svg@normalize@@path#1/#2/\@nil{%
  \IfArgIsEmpty{#2}{%
    \IfArgIsEmpty{#1}{\def\svg@tempa{}}{\def\svg@tempa{#1/}}%
  }{%
    \svg@normalize@@path#2/\@nil%
    \edef\svg@tempa{#1/\svg@tempa}%
  }%
}
```

A.3.4. List handling

`\svgx@ifinlist`

Check, if the first argument is included in a comma-separated list in the second argument. Keep in mind that the first argument is not expanded at all, the second one exactly once.

```
\newcommand*\svgx@ifinlist[2]{%
  \begingroup%
    \def\svg@tempa##1,#1,##2\@nil{%
      \IfArgIsEmpty{##2}{%
        \aftergroup\@secondoftwo%
      }{%
        \aftergroup\@firstoftwo%
      }%
    }%
    \expandafter\svg@tempa\expandafter,#2,#1,\@nil%
  \endgroup%
}
```

B. Including SVG files with package svg

B.1. Options

All options, which are recommended to be set with `\svgsetup{`⟨*options*⟩`}` but are also available as package options, as well as the optional parameters for both user commands `\includesvg[`⟨*parameters*⟩`]{`⟨*svg file*⟩`}` and `\includeinkscape[`⟨*parameters*⟩`]{`⟨*file*⟩`}` are defined with the interface provided by package **scrbase**.

```
\DefineFamily{SVG}
\DefineFamilyMember{SVG}
```

`\svg@deprecated@key`

With version v2.00 the whole user-interface was renewed. For reasons of compatibility, outdated options and parameters from version v1.0 are also provided. If an old key was given, a warning is issued and the valid key is used.

```
\newcommand*\svg@deprecated@key[3][svg]{%
  \PackageWarning{#1}{%
    The option key ‘#2’ is deprecated. \MessageBreak%
    It’s recommended to use ‘#3’\MessageBreak%
    instead%
  }%
  \FamilyOptions{SVG}{#3}%
}
```

Within the exported LaTeX files of ***Inkscape***, some commands are used out of additional packages. But maybe the user doesn't want to load this packages anyhow.

`usexcolor` (opt.)
`noxcolor` (opt.)
`\if@svg@use@xcolor`
`usetransparent` (opt.)
`notransparent` (opt.)
`\if@svg@use@transparent`

Options for preventing packages **xcolor** and **transparent** to be loaded.

```
\newif\if@svg@use@xcolor
\FamilyBoolKey{SVG}{usexcolor}{@svg@use@xcolor}
\DeclareOption{noxcolor}{\FamilyOptions{SVG}{usexcolor=false}}
\newif\if@svg@use@transparent
\FamilyBoolKey{SVG}{usetransparent}{@svg@use@transparent}
\DeclareOption{notransparent}{\FamilyOptions{SVG}{usetransparent=false}}
```

They are only available during the loading process of package **svg**.

```
\AtEndOfPackage{%
  \RelaxFamilyKey{SVG}{usexcolor}%
  \RelaxFamilyKey{SVG}{usetransparent}%
  \if@svg@use@xcolor%
    \RequirePackage{xcolor}[2016/05/11]%
  \else%
    \AfterPackage*{xcolor}{%
      \PackageWarning{svg}{Package `xcolor' was loaded anyway}%
    }%
  \fi%
  \if@svg@use@transparent%
    \RequirePackage{transparent}[2019/11/29]%
  \else%
    \AfterPackage*{transparent}{%
      \PackageWarning{svg}{Package `transparent' was loaded anyway}%
    }%
  \fi%
```

There is an issue with package **transparent**, which currently implements an *invalid* check relying on internal commands of package **pgfsys**, whereas these have changed in the latest version.[11]

```
  \AfterPackage*{transparent}{%
    \ifcsname Gin@driver\endcsname%
      \RequirePackage{pgfsys}%
    \fi%
  }%
}
```

B.1.1. The invocation of *Inkscape*

The Application ***Inkscape*** is used to create includable graphic files in a desired format (PDF/EPS/PS/PNG) out of files in SVG format, whereas the support of LaTeX can optionally be used.

`inkscape` (opt.)
`\svg@ink@mode`

The intension of option `inkscape` is to control the running behaviour of ***Inkscape***. It can be switched off at all (`inkscape=false`) or invoked only if necessary (`inkscape=true`) and even be forced with every LaTeX run (`inkscape=forced`). Additionally, option `inkscape` can be used as wrapper for options `inkscapeformat`, `inkscapelatex`, `inkscapearea` and `inkscapedpi`, which are declared later.

```
\newcommand*\svg@ink@mode{}
\DefineFamilyKey{SVG}{inkscape}[true]{%
  \svg@sanitize@dq\svg@tempb{#1}%
```

---

[11] https://github.com/ho-tex/transparent/issues/3

```
\FamilySetNumerical{SVG}{inkscape}{svg@tempa}{%
  {false}{0},{off}{0},{no}{0},%
  {true}{1},{on}{1},{yes}{1},{auto}{1},{onlynewer}{1},{newer}{1},%
  {forced}{2},{force}{2},{overwrite}{2},%
  {pdf}{3},{PDF}{3},{eps}{4},{EPS}{4},{ps}{5},{PS}{5},{png}{6},{PNG}{6},%
  {drawing}{7},{crop}{7},%
  {page}{8},{nocrop}{8},%
  {tex}{9},{latex}{9},{exportlatex}{9},{latexexport}{9},%
  {notex}{10},{nolatex}{10},{noexportlatex}{10},{nolatexexport}{10},%
  {latexnoexport}{10},{raw}{10},{plain}{10},{simple}{10}%
}{\svg@tempb}%
\ifx\FamilyKeyState\FamilyKeyStateProcessed%
```

Setting the mode for invoking ***Inkscape***…

```
  \ifnum\svg@tempa<\thr@@\relax%
    \let\svg@ink@mode\svg@tempa%
  \else%
```

…and the part as wrapper for different options.

```
    \ifcase\svg@tempa\relax\or\or\or% pdf
      \FamilyOptions{SVG}{inkscapeformat=pdf}%
    \or% eps
      \FamilyOptions{SVG}{inkscapeformat=eps}%
    \or% ps
      \FamilyOptions{SVG}{inkscapeformat=ps}%
    \or% png
      \FamilyOptions{SVG}{inkscapeformat=png}%
    \or% drawing
      \FamilyOptions{SVG}{inkscapearea=drawing}%
    \or% page
      \FamilyOptions{SVG}{inkscapearea=page}%
    \or% tex
      \FamilyOptions{SVG}{inkscapelatex=true}%
    \or% notex
      \FamilyOptions{SVG}{inkscapelatex=false}%
    \fi%
  \fi%
```

It's also possible to set the option `inkscapedpi` by passing a number followed by `dpi` like `inkscape=300dpi`.

```
\else% dpi
  \def\svg@tempa##1dpi##2\@nil{%
    \Ifstr{##2}{dpi}{\FamilyOptions{SVG}{inkscapedpi=##1}}{}%
  }%
  \lowercase{\expandafter\svg@tempa\svg@tempb dpi\@nil}%
```

In version v1.0 the option `inkscape` was used to set both the executable and options for ***Inkscape***. This is taken into account here.

```
  \ifx\FamilyKeyState\FamilyKeyStateProcessed\else% legacy option
```

Splitting executable from options with delimited macros. After calling `\svg@tempa` with the given value, the part for the executable is stored in `\svg@tempa` and the option part—which is recognized by the first `-` character— in `\svg@tempb`.

```
    \svg@quotes@remove[{#1}]{\svg@tempb}%
    \def\svg@tempa##1-##2\@nil{%
      \IfArgIsEmpty{##2}{\let\svg@tempb\@empty}{%
        \def\svg@tempa####1-\@nil{\def\svg@tempb{-####1}}%
        \svg@tempa##2\@nil%
      }%
      \edef\svg@tempa{\trim@spaces{##1}}%
    }%
    \edef\svg@tempb{%
```

```
        \noexpand\svg@tempa\svg@tempb-\noexpand\@nil%
      }%
      \svg@tempb%
      \if@svg@quotes@found%
        \edef\svg@tempa{"\svg@tempa"}%
      \fi%
      \PackageWarning{svg}{%
        Setting the executable%
        \ifx\svg@tempb\@empty\else%
          \space and associated options%
        \fi%
        \MessageBreak%
        for Inkscape should be done with options\MessageBreak%
        ‘inkscapeexe=\svg@tempa’%
        \ifx\svg@tempb\@empty\else%
          \MessageBreak and ‘inkscapeopt=\svg@tempb’%
        \fi.\MessageBreak%
        Nevertheless, this was done by now anyway%
      }%
      \edef\svg@tempa{%
        \noexpand\FamilyOptions{SVG}{inkscapeexe=\svg@tempa}%
        \ifx\svg@tempb\@empty\else%
          \noexpand\FamilyOptions{SVG}{inkscapeopt=\svg@tempb}%
        \fi%
      }%
      \svg@tempa%
    \fi%
  \fi%
}
```

`on` (opt.)
`off` (opt.)

Package options which can be used to switch functionality on or off during the loading of package **svg**.

```
\DeclareOption{on}{\FamilyOptions{SVG}{inkscape=true}}
\DeclareOption{off}{\FamilyOptions{SVG}{inkscape=false}}
```

`inkscapeversion` (opt.)
`\svg@ink@ver`
`inkscapeexe` (opt.)
`\svg@ink@exe`
`inkscapeopt` (opt.)
`\svg@ink@opt`

With these options, the executed command for invoking ***Inkscape*** as well as additional options can be defined.

```
\newcommand*\svg@ink@ver{\m@ne}
\DefineFamilyKey{SVG}{inkscapeversion}[true]{%
  \FamilySetNumerical{SVG}{inkscape}{svg@tempa}{%
    {true}{0},{on}{0},{yes}{0},{auto}{0},{detect}{0},{determine}{0},{fetch}{0},%
    {enquire}{0},{identify}{0},{request}{0},{retrieve}{0},{obtain}{0}%
  }{#1}%
  \ifx\FamilyKeyState\FamilyKeyStateProcessed%
    \renewcommand*\svg@ink@ver{\m@ne}%
  \else%
    \def\svg@tempa##1.##2\@nil{%
      \Ifnumber{##1}{%
        \renewcommand*\svg@ink@ver{##1}%
        \FamilyKeyStateProcessed%
      }{}%
    }%
    \svg@tempa#1.\@nil%
  \fi%
}
\newcommand*\svg@ink@exe{inkscape}
\DefineFamilyKey{SVG}{inkscapeexe}{%
  \svg@sanitize@dq\svg@ink@exe{#1}%
  \FamilyKeyStateProcessed%
}
\newcommand*\svg@ink@opt{}
\DefineFamilyKey{SVG}{inkscapeopt}{%
  \renewcommand*\svg@ink@opt{#1}%
  \FamilyKeyStateProcessed%
}
```

The two options `inkscapeversion` and `inkscapeexe` can only be used within the preamble.

```
\def\svg@tempa#1{%
  \AtBeginDocument{%
    \DefineFamilyKey[]{SVG}{#1}[]{%
      \PackageError{svg}{Option ‘#1’ too late}{%
        Option ‘#1’ can only be set within\MessageBreak%
        the preamble but you have tried to set it up later.%
      }%
      \FamilyKeyStateProcessed%
    }%
  }%
}
\svg@tempa{inkscapeexe}
\svg@tempa{inkscapeversion}
```

`inkscapeformat` (opt.)
`\svg@ink@format`

With option `inkscapeformat` the output format of the ***Inkscape*** export function, which is called via `\ShellEscape`, can be configured. It is set to `pdf` or, if dvi output could be detected, to `eps` during initialization.

```
\newcommand*\svg@ink@format{pdf}
\ifxetex\else\ifpdf\else
  \renewcommand*\svg@ink@format{eps}
\fi\fi
\DefineFamilyKey{SVG}{inkscapeformat}{%
  \FamilySetNumerical{SVG}{inkscapeformat}{svg@tempa}{%
    {pdf}{0},{PDF}{0},{eps}{1},{EPS}{1},{ps}{2},{PS}{2},{png}{3},{PNG}{3}%
  }{#1}%
  \ifx\FamilyKeyState\FamilyKeyStateProcessed%
    \ifcase\svg@tempa\relax% latex
      \renewcommand*\svg@ink@format{pdf}%
    \or% eps
      \renewcommand*\svg@ink@format{eps}%
    \or% ps
      \renewcommand*\svg@ink@format{ps}%
    \or% png
      \renewcommand*\svg@ink@format{png}%
    \fi%
  \fi%
}
```

`inkscapelatex` (opt.)
`latex` (opt.)
`tex` (opt.)
`\svg@ink@latex`

This option controls whether the ***Inkscape*** export will be invoked with or without the generation of a separate LaTeX file.

```
\newif\if@svg@ink@latex
\FamilyBoolKey{SVG}{inkscapelatex}{@svg@ink@latex}
\FamilyBoolKey{SVG}{latex}{@svg@ink@latex}
\FamilyBoolKey{SVG}{tex}{@svg@ink@latex}
```

`inkscapearea` (opt.)
`\svg@ink@area`

The exported area for an ***Inkscape*** graphic can be set with this option.

```
\newcommand*\svg@ink@area{}
\DefineFamilyKey{SVG}{inkscapearea}{%
  \FamilySetNumerical{SVG}{inkscapearea}{svg@tempa}{%
    {drawing}{0},{crop}{0},%
    {page}{1},{nocrop}{1}%
  }{#1}%
  \ifx\FamilyKeyState\FamilyKeyStateProcessed%
    \ifcase\svg@tempa\relax% drawing
      \renewcommand*\svg@ink@area{-D}%
    \else% page
      \renewcommand*\svg@ink@area{-C}%
    \fi%
  \fi%
}
```

`inkscapedpi` (opt.)
`inkscapedensity` (opt.)
`\svg@ink@dpi`

A density can be chosen, which is used during export with ***Inkscape*** for bitmaps and rasterization of filters.

```
\newcommand*\svg@ink@dpi{}
\let\svg@ink@dpi\relax
\DefineFamilyKey{SVG}{inkscapedpi}{%
  \FamilyKeyStateUnknownValue%
  \svg@ifvalueisrelax{#1}{%
    \let\svg@ink@dpi\relax%
    \FamilyKeyStateProcessed%
  }{%
    \def\svg@tempa##1dpi##2\@nil{\def\svg@tempa{##1}}%
    \lowercase{\svg@tempa#1dpi\@nil}%
    \Ifnumber{\svg@tempa}{%
      \edef\svg@ink@dpi{\svg@tempa}%
      \FamilyKeyStateProcessed%
    }{}%
  }%
}
\DefineFamilyKey{SVG}{inkscapedensity}{\FamilyOptions{SVG}{inkscapedpi=#1}}
```

`\svg@ink@cmd`

The actual usage of the ***Inkscape*** command line interface.

```
\newcommand*\svg@ink@cmd[2]{%
  \svg@ink@exe\space"#1.\svg@file@ext"\space\svg@ink@area\space%
  \ifx\svg@ink@dpi\relax\else--export-dpi=\svg@ink@dpi\space\fi%
  \if@svg@ink@latex--export-latex\space\fi%
  \ifx\svg@ink@opt\@empty\else\svg@ink@opt\space\fi%
  \ifcase\svg@ink@ver\relax% 0.x detected
    --without-gui\space%
    --export-\svg@ink@format="#2.\svg@ink@format"%
  \else% 1.x or nothing detected
    --export-filename="#2.\svg@ink@format"%
  \fi%
}
```

B.1.2. Setting input folder and file

`svgpath` (opt.)

In version v1.0 setting the path to SVG files was done via option. So this method is provided as well.

```
\DefineFamilyKey{SVG}{svgpath}{%
  \PackageWarning{svg}{%
    The key ‘svgpath’ is deprecated. It’s recommended\MessageBreak%
    to use ‘\string\svgpath’ instead%
  }%
  \ifx\svgpath\@undefined%
    \AtEndOfPackage{\svgpath{{#1}}}%
  \else%
    \svgpath{{#1}}%
  \fi%
  \FamilyKeyStateProcessed%
}
```

`svgextension` (opt.)
`extension` (opt.)
`ext` (opt.)
`\svg@file@ext`

This option modifies the expected extension for the input file which is exported with ***Inkscape***. It is set to `svg` by default.

```
\newcommand*\svg@file@ext{svg}
\DefineFamilyKey{SVG}{svgextension}{%
```

The extension should be in lower case letters.

```
  \lowercase{\svg@quotes@remove[{#1}]{\svg@file@ext}}%
```

Remove leading dots from the extension.

```
  \svg@remove@leadingchar.\svg@file@ext%
}
\DefineFamilyKey{SVG}{extension}{\FamilyOptions{SVG}{svgextension=#1}}
\DefineFamilyKey{SVG}{ext}{\FamilyOptions{SVG}{svgextension=#1}}
```

B.1.3. Setting output folder and file

`inkscapepath` (opt.)
`\svg@out@path`

The option `inkscapepath` controls, in which folder the results of the ***Inkscape*** export will be located.

```
\newcommand*\svg@out@path{}
\DefineFamilyKey{SVG}{inkscapepath}{%
  \svg@sanitize@dq\svg@tempb{#1}%
  \FamilySetNumerical{SVG}{inkscapepath}{svg@tempa}{%
    {svgpath}{0},{svgdir}{0},%
    {svgsubpath}{1},{svgsubdir}{1},%
    {basepath}{2},{basedir}{2},{jobpath}{2},{jobdir}{2},%
    {basesubpath}{3},{basesubdir}{3},{jobsubpath}{3},{jobsubdir}{3}%
  }{\svg@tempb}%
  \ifx\FamilyKeyState\FamilyKeyStateProcessed%
    \ifcase\svg@tempa\relax% svgpath
      \renewcommand*\svg@out@path{\svg@file@path}%
    \or% svgsubpath
      \renewcommand*\svg@out@path{\svg@file@path svg-inkscape/}%
    \or% basepath
      \renewcommand*\svg@out@path{./}%
    \or% basesubpath
      \renewcommand*\svg@out@path{./svg-inkscape/}%
    \fi%
  \else%
    \edef\svg@out@path{\svg@tempb}%
    \svg@normalize@path{\svg@out@path}%
    \FamilyKeyStateProcessed%
  \fi%
}
```

`inkscapename` (opt.)
`\svg@out@name`
`\svg@out@base`

With option `inkscapename` the name of the exported file can be changed.

```
\newcommand*\svg@out@name{\svg@file@name\svg@file@suffix}
\newcommand*\svg@out@base{\svg@out@path\svg@out@name.\svg@ink@format}
\DefineFamilyKey{SVG}{inkscapename}{%
  \renewcommand*\svg@out@name{#1\svg@file@suffix}%
  \FamilyKeyStateProcessed%
}
```

B.1.4. Options for the inclusion of graphics

After the graphic export with ***Inkscape***, the inclusion of those graphics can be controlled with the following options.

`width` (opt.)
`\svg@param@width`
`height` (opt.)
`\svg@param@width`
`distort` (opt.)
`keepaspectratio` (opt.)
`\if@svg@param@distort`
`scale` (opt.)
`\svg@param@scale`

These options determine the size of the included graphics. The usage of `\relax` as value resets the respective option to the default behavior.

```
\newcommand*\svg@param@width{\z@}
\DefineFamilyKey{SVG}{width}{%
  \FamilyKeyStateUnknownValue%
  \svg@ifvalueisrelax{#1}{%
```

```
    \renewcommand*\svg@param@width{\z@}%
    \FamilyKeyStateProcessed%
  }{%
    \FamilySetLengthMacro{SVG}{width}{\svg@param@width}{#1}%
    \ifx\FamilyKeyState\FamilyKeyStateProcessed%
      \ifdim\svg@param@width<\z@\relax%
        \FamilyKeyStateUnknownValue%
      \fi%
    \fi%
  }%
}
\newcommand*\svg@param@height{\z@}
\DefineFamilyKey{SVG}{height}{%
  \FamilyKeyStateUnknownValue%
  \svg@ifvalueisrelax{#1}{%
    \renewcommand*\svg@param@height{\z@}%
    \FamilyKeyStateProcessed%
  }{%
    \FamilySetLengthMacro{SVG}{height}{\svg@param@height}{#1}%
    \ifx\FamilyKeyState\FamilyKeyStateProcessed%
      \ifdim\svg@param@height<\z@\relax%
        \FamilyKeyStateUnknownValue%
      \fi%
    \fi%
  }%
}
\newif\if@svg@param@distort
\FamilyBoolKey{SVG}{distort}{@svg@param@distort}
\DefineFamilyKey{SVG}{keepaspectratio}[true]{%
  \FamilySetBool{SVG}{keepaspectratio}{@svg@tempswa}{#1}%
  \ifx\FamilyKeyState\FamilyKeyStateProcessed%
    \if@svg@tempswa%
      \FamilyExecuteOptions[.svg.sty]{SVG}{distort=false}%
    \else%
      \FamilyExecuteOptions[.svg.sty]{SVG}{distort=true}%
    \fi%
  \fi%
}
\newcommand*\svg@param@scale{1}
\DefineFamilyKey{SVG}{scale}{%
  \FamilyKeyStateUnknownValue%
  \svg@ifvalueisrelax{#1}{%
    \renewcommand*\svg@param@scale{1}%
    \FamilyKeyStateProcessed%
  }{%
    \Ifisdimension{#1\p@}{%
      \ifdim\dimexpr#1\p@\relax>\z@\relax%
        \renewcommand*\svg@param@scale{#1}%
        \FamilyKeyStateProcessed%
      \fi%
    }{}%
  }%
}
```

`pretex` (opt.)
`\svg@param@pretex`
`apptex` (opt.)
`\svg@param@apptex`
`postex` (opt.)

For executing code right before or after the graphic inclusion, two hooks are defined.

```
\newcommand*\svg@param@pretex{}
\let\svg@param@pretex\relax
\DefineFamilyKey{SVG}{pretex}{%
  \svg@ifvalueisrelax{#1}{%
    \let\svg@param@pretex\relax%
  }{%
    \def\svg@param@pretex{#1}%
  }%
  \FamilyKeyStateProcessed%
}
\newcommand*\svg@param@apptex{}
\let\svg@param@apptex\relax
\DefineFamilyKey{SVG}{apptex}{%
  \svg@ifvalueisrelax{#1}{%
    \let\svg@param@apptex\relax%
  }{%
    \def\svg@param@apptex{#1}%
  }%
  \FamilyKeyStateProcessed%
}
\DefineFamilyKey{SVG}{postex}{%
  \svg@deprecated@key{postex=#1}{apptex=#1}%
}
```

`lastpage` (opt.)
`svg@param@lastpage` (counter)

For ***Inkscape*** 0.91 a bug concerning the LaTeX export has been reported (https://bugs.launchpad.net/ubuntu/+source/inkscape/+bug/1417470). Sometimes the LaTeX file created by ***Inkscape*** tries to include more pages than actually are present in the PDF file. To work around this problem, a patch is provided. For this purpose, the total page number is read from the PDF file.

```
\newcounter{svg@param@lastpage}
\DefineFamilyKey{SVG}{lastpage}[true]{%
  \FamilySetNumerical{SVG}{lastpage}{svg@tempa}{%
    {false}{0},{off}{0},{no}{0},{ignore}{0},%
    {true}{1},{on}{1},{yes}{1},{auto}{1}%
  }{#1}%
  \ifx\FamilyKeyState\FamilyKeyStateProcessed%
    \ifcase\svg@tempa\relax% false
      \FamilySetCounter{SVG}{lastpage}{svg@param@lastpage}{\m@ne}%
    \or% true
      \FamilySetCounter{SVG}{lastpage}{svg@param@lastpage}{\z@}%
    \fi%
  \fi%
}
```

`draft` (opt.)
`\if@svg@draft`

The option `draft` has the same effect as the eponymous option of package **graphicx**.

```
\newif\if@svg@draft
\FamilyBoolKey{SVG}{draft}{@svg@draft}
\AfterPackage*{graphicx}{\ifGin@draft\@svg@drafttrue\fi}
```

B.2. User commands

B.2.1. Optional parameters for user commands

The family member is defined for both **svg** and **svg-extract**.

```
⟨*package & body⟩
\DefineFamilyMember[.param]{SVG}
⟨/package & body⟩
```

`\svg@local@param@def`
`\svg@local@param@use`
`\svg@local@param@set`

Most of the package options can also be used as optional parameters for `\includesvg` or `\includeinkscape`. Some of them are overloaded for the usage as optional argument and there are some keys, which *only* can be used as optional parameters. This is realized in such a way that `\svg@local@param@use` is extended with `\svg@local@param@def` by the definition of local keys during the loading of package **svg**.

```
\newcommand*\svg@local@param@use{}
\newcommand*\svg@local@param@def[1]{%
  \edef\svg@local@param@use{%
    \unexpanded\expandafter{\svg@local@param@use}\unexpanded{#1}%
  }%
```

```
}
\newcommand*\svg@local@param@set[1]{%
  \svg@local@param@use%
  \FamilyOptions{SVG}{#1}%
```

As `\svg@local@param@set` is always used in a local group, it is possible to set `inkscapelatex` to `false`, if the output format was set to `png` with option `inkscapeformat`.

```
  \Ifstr{\svg@ink@format}{png}{\FamilyOptions{SVG}{inkscapelatex=false}}{}%
```

Using `distort=true` is only reasonable, if `height` and `width` are given.

```
  \@svg@tempswatrue%
  \ifdim\svg@param@width>\z@\relax\ifdim\svg@param@height>\z@\relax%
    \@svg@tempswafalse%
  \fi\fi%
  \if@svg@tempswa%
    \FamilyExecuteOptions[.svg.sty]{SVG}{distort=false}%
  \fi%
}
```

`\svg@deprecated@param`

This macro checks, if `\svgwidth` or `\svgscale` are defined. In this case, the given values are passed to the correlating parameters and a warning is raised.

```
\newcommand*\svg@deprecated@param{%
  \@svg@tempswafalse%
  \ifx\svgwidth\@undefined\else%
    \edef\svg@tempa{\noexpand\FamilyOptions{SVG}{width=\svgwidth}}%
    \svg@tempa%
    \@svg@tempswatrue%
  \fi%
  \ifx\svgscale\@undefined\else%
    \edef\svg@tempa{\noexpand\FamilyOptions{SVG}{scale=\svgscale}}%
    \svg@tempa%
    \@svg@tempswatrue%
  \fi%
  \if@svg@tempswa%
    \PackageWarning{svg}{%
      You should specify the image size with parameters\MessageBreak%
      ‘width’ and ‘height’ or ‘scale’ instead of using\MessageBreak%
      ‘\string\svgscale’ or ‘\string\svgwidth’%
    }%
    \let\svgwidth\@undefined%
    \let\svgscale\@undefined%
  \fi%
}
```

B.2.2. Definition of user commands

`\svgsetup`
`\setsvg`

The macro `\svgsetup{`⟨*options*⟩`}` can be used to change options after loading packages **svg** or **svg-extract** both in preamble and the document body. For compatibility reasons, `\setsvg` is also defined.

```
\newcommand*\svgsetup{\FamilyOptions{SVG}}
\newcommand*\setsvg{\FamilyOptions{SVG}}
```

`\svgpath`
`\svg@input@path`

With `\svgpath` the user can give several root paths to SVG files in the same way as `\graphicspath` is used. The only difference is that a missing slash is added at the end of the path, if needed.

```
\newcommand*\svg@input@path{}
\let\svg@input@path\input@path
\newcommand*\svgpath[1]{%
  \def\svg@tempa##1\@nil{%
```

```
    \ifx\svg@tempb\bgroup%
      \def\svg@input@path{#1}%
    \else%
      \def\svg@input@path{{#1}}%
    \fi%
  }%
  \futurelet\svg@tempb\svg@tempa#1\@nil%
}
```

`\includesvg`

For the inclusion of SVG files the command `\includesvg` is defined.

```
\newcommand*\includesvg[2][]{%
  \begingroup%
```

Checking for deprecated commands `\svgwidth` and `\svgscale`.

```
    \svg@deprecated@param%
```

`inkscape` (param.)
`inkscapeformat` (param.)
`inkscapelatex` (param.)
`inkscapearea` (param.)
`inkscapedpi` (param.)
`inkscapeopt` (param.)
`svgextension` (param.)
`width` (param.)
`height` (param.)
`distort` (param.)
`scale` (param.)
`pretex` (param.)
`apptex` (param.)
`draft` (param.)

Most of the optional parameters have the same effect as the identically named options. Only parameter `lastpage` is extended (see below). Moreover, there are some additional parameters, which can only be used as optional argument for `\includesvg` (`angle` and `origin`) but not as an option. Now all parameters are set in local context (within a group).

```
    \svg@local@param@set{#1}%
```

The file suffix used by both packages **svg** and **svg-extract**.

```
    \if@svg@ink@latex%
      \edef\svg@file@suffix{_\svg@file@ext-tex}%
    \else%
      \edef\svg@file@suffix{_\svg@file@ext-raw}%
    \fi%
    \@onelevel@sanitize\svg@file@suffix%
```

Searching all given paths for the relevant SVG file.

```
    \svg@get@path{#2}{}%
    \if@svg@file@found%
```

Running the export with ***Inkscape*** (if necessary) and checking the required files for graphic inclusion.

```
      \svg@ink@run%
      \IfFileExists{\svg@out@base}{}{%
        \@svg@file@foundfalse%
        \svg@file@missing{\svg@out@base}{\svg@file@base.\svg@file@ext}%
      }%
      \if@svg@ink@latex%
        \IfFileExists{\svg@out@base_tex}{}{%
          \@svg@file@foundfalse%
          \svg@file@missing{\svg@out@base_tex}{\svg@file@base.\svg@file@ext}%
        }%
      \fi%
```

Include the resulting graphic file and maybe extract independent files.

```
      \if@svg@file@found%
        \svg@input{\svg@out@base}%
        \svg@extract{\svg@out@base}%
      \fi%
    \else%
```

Raise an error, if the requested SVG file wasn't found.

```
    \svg@file@missing[\svg@file@ext]{\svg@file@base}{}%
  \fi%
  \endgroup%
}
```

`lastpage` (param.)

In addition to the automatic finding of the last page, which is included, it can also be given directly as parameter.

```
\svg@local@param@def{%
  \FamilyCounterKey[.param]{SVG}{lastpage}{svg@param@lastpage}%
}
```

`angle` (param.)
`origin` (param.)

The parameters `angle` and `origin` are defined as pendants to the keys provided by `\includegraphics`.

```
\newcommand*\svg@param@angle{0}
\svg@local@param@def{%
  \DefineFamilyKey[.param]{SVG}{angle}{%
    \FamilyKeyStateUnknownValue%
    \Ifisdimension{#1\p@}{%
      \renewcommand*\svg@param@angle{#1}%
      \FamilyKeyStateProcessed%
    }{}%
  }%
}
\newcommand*\svg@param@origin{c}
\svg@local@param@def{%
  \DefineFamilyKey[.param]{SVG}{origin}[c]{%
    \renewcommand*\svg@param@origin{#1}%
    \FamilyKeyStateProcessed%
  }%
}
```

`\includeinkscape`

The command `\includeinkscape` can be used for including the export results of ***Inkscape***, if this part of the job was done in another way.

```
\newcommand*\includeinkscape[2][]{%
  \begingroup%
```

Checking for deprecated commands `\svgwidth` and `\svgscale`.

```
    \svg@deprecated@param%
```

The given file extension is examined, where a known extension overwrites the current setting for `inkscapeformat`. If there's a suffix `_tex`, the option `inkscapelatex` is set to `true` by default.

```
    \svg@filename@parse{#2}%
    \ifx\filename@ext\relax\else%
      \svg@quotes@remove{\filename@ext}%
      \expandafter\lowercase\expandafter{%
        \expandafter\def\expandafter\filename@ext\expandafter{\filename@ext}%
      }%
      \def\svg@tempb##1_tex##2\@nil{%
        \IfArgIsEmpty{##1}{}{\def\filename@ext{##1}}%
        \Ifstr{##2}{_tex}{\@svg@tempswatrue}{\@svg@tempswafalse}%
      }%
      \@svg@tempswafalse%
      \@tfor\svg@tempa:={pdf}{eps}{ps}{png}\do{%
        \begingroup%
          \expandafter\svg@tempb\filename@ext_tex\@nil%
          \svg@extension@parse{\svg@tempa}%
          \ifx\filename@ext\relax%
```

```
            \def\svg@tempb{\endgroup}%
          \else%
            \edef\svg@tempb{%
              \endgroup%
              \noexpand\FamilyOptions{SVG}{inkscapeformat=\svg@tempa}%
              \if@svg@tempswa%
                \noexpand\FamilyOptions{SVG}{inkscapelatex=true}%
              \fi%
              \def\noexpand\filename@base{\filename@base}%
              \def\noexpand\filename@ext{\filename@ext}%
              \noexpand\@svg@tempswatrue%
            }%
          \fi%
        \svg@tempb%
```

Break for loop, if valid extension was found.

```
        \if@svg@tempswa%
          \@break@tfor%
        \fi%
      }%
```

If no valid extension was found, it is set to the specified format and the actual found one is appended to cssvg.dtx@base.

```
      \if@svg@tempswa\else%
        \svg@extension@parse{\svg@ink@format}%
      \fi%
    \fi%
```

`inkscapeformat` (param.)
`inkscapelatex` (param.)
`width` (param.)
`height` (param.)
`distort` (param.)
`scale` (param.)
`pretex` (param.)
`apptex` (param.)
`draft` (param.)
`lastpage` (param.)
`angle` (param.)
`origin` (param.)

Parameters, which are supported by `\includesvg`, can also be used with `\includeinkscape` even if some of them—more precisely those that control the export with ***Inkscape***—don't have an effect at all. Nevertheless, they are set right now in local context (within a group).

```
    \svg@local@param@set{#1}%
```

Searching all given paths for the relevant PDF/EPS file.

```
    \svg@get@path[\svg@ink@format]{\filename@area\filename@base}{\svg@out@path}%
    \if@svg@file@found%
```

Checking the required files for graphic inclusion.

```
      \edef\svg@out@name{\svg@file@name}%
      \edef\svg@out@base{\svg@file@path\svg@file@name.\svg@ink@format}%
      \if@svg@ink@latex%
        \IfFileExists{\svg@out@base_tex}{}{%
          \@svg@file@foundfalse%
          \svg@file@missing{\svg@out@base_tex}{\svg@out@base}%
        }%
      \fi%
```

Include the resulting graphic file and maybe extract independent files.

```
      \if@svg@file@found%
        \svg@input{\svg@out@base}%
        \svg@extract{\svg@out@base}%
      \fi%
    \else%
```

Raise an error, if the requested PDF/EPS file wasn't found.

```
      \svg@file@missing[\svg@ink@format]{\svg@file@base}{\svg@out@path}%
    \fi%
  \endgroup%
}
```

B.3. Auxiliary macros

`\svg@ink@run`
`\if@svg@ink@run`

The command, which performs the call of ***Inkscape*** via `\ShellEscape`.

```
\newif\if@svg@ink@run
\newcommand*\svg@ink@run{%
  \ifnum\svg@ink@mode>\z@\relax%
    \begingroup%
```

If the mode for `inkscape` was set to `forced`, ***Inkscape*** will be called in any case. Otherwise, some checks are performed to detect, if a run of ***Inkscape*** is actually necessary.

```
      \@svg@ink@runtrue%
      \ifnum\svg@ink@mode=\tw@\relax\else%
```

This is the case when the SVG file is newer than the corresponding exported file, or if the latter isn't present at all.

```
        \svg@iffilenewer{\svg@file@base.\svg@file@ext}{\svg@out@base}{}{%
          \@svg@ink@runfalse%
        }%
```

The same is true, when the associated LaTeX file is missing. But when this file already exists, maybe the user did some changes to this file. In this case, overwriting this file is maybe not intended.

```
        \if@svg@ink@latex%
          \IfFileExists{\svg@out@base_tex}{%
            \ifnum\pdf@shellescape=\@ne\relax\if@svg@ink@run%
              \svg@iffilenewer{\svg@out@base_tex}{\svg@out@base}{%
                \@svg@ink@runfalse%
                \svg@quotes@remove[\svg@out@base]{\svg@tempa}%
                \PackageWarning{svg}{%
                  Since the encountered filedate of file\MessageBreak%
                  ‘\svg@tempa_tex’ is newer than \MessageBreak%
                  ‘\svg@tempa’ it’s supposed that\MessageBreak%
                  you customized this file. To avoid an accidental\MessageBreak%
                  overwriting of this file, the Inkscape export\MessageBreak%
                  won’t be done. If you want to overwrite the\MessageBreak%
                  existing file please choose the parameter\MessageBreak%
                  ‘inkscape=force’%
                }%
              }{}%
            \fi\fi%
          }{\@svg@ink@runtrue}%
        \fi%
      \fi%
```

If all checks were positive, the export with ***Inkscape*** can be done in case flag `--shell-escape` is used.

```
      \if@svg@ink@run%
        \ifnum\pdf@shellescape=\@ne\relax%
```

For exporting PNG files, the used density is set to `300dpi`, if no value was given.

```
          \ifx\svg@ink@dpi\relax%
            \Ifstr{\svg@ink@format}{png}{%
              \FamilyOptions{SVG}{inkscapedpi=300}%
            }{}%
          \fi%
          \PackageInfo{svg}{%
            Calling Inkscape%
            \ifx\svg@ink@opt\@empty\else%
              \space with added options ‘\svg@ink@opt’%
            \fi%
          }%
```

Executing ***Inkscape*** on shell. Afterwards, the export results are moved into the given output path.

```
          \svg@quotes@remove[\svg@file@base]{\svg@tempa}%
          \svg@quotes@remove[\svg@out@name]{\svg@tempb}%
```

The last try to detect the version automatically, if this wasn't successful until now.

```
          \ifnum\svg@ink@ver=\m@ne\relax%
            \svg@ink@ver@explore{\svg@tempa}{\svg@tempb}{\svg@out@name}%
          \fi%
```

Now it's time to actually create the desired graphic.

```
          \ShellEscape{\svg@ink@cmd{\svg@tempa}{\svg@tempb}}%
          \IfFileExists{\svg@out@name.\svg@ink@format}{%
            \edef\svg@tempb{\svg@tempb.\svg@ink@format}%
            \svg@quotes@remove{\svg@out@base}%
            \svg@shell@mkdir{\svg@out@path}%
            \svg@shell@mv{\svg@tempb}{\svg@out@base}%
            \if@svg@ink@latex%
              \svg@shell@mv{\svg@tempb_tex}{\svg@out@base_tex}%
            \fi%
          }{%
            \gdef\svg@ink@ver{\m@ne}%
            \PackageWarning{svg}{%
              The export with Inkscape failed for file\MessageBreak%
              `\svg@tempa.\svg@file@ext'\MessageBreak%
              Troubleshooting: Please check in the log file how\MessageBreak%
              the invocation of Inkscape took place and try to\MessageBreak%
              execute it yourself in the terminal%
            }%
          }%
```

If `--shell-escape` wasn't enabled, a warning is issued.

```
        \else%
          \svg@quotes@remove[\svg@file@base]{\svg@tempa}%
          \PackageWarning{svg}{%
            You didn't enable `shell escape' (or `write18')\MessageBreak%
            so it wasn't possible to launch the Inkscape export\MessageBreak%
            for `\svg@tempa.\svg@file@ext'%
          }%
        \fi%
      \fi%
    \endgroup%
  \fi%
}
```

`\svg@input`
`\svg@@input`
`\svg@box`

With `\svg@@input` the export results of ***Inkscape*** are included. The macro `\svg@input` is defined in order to realize the option `exclude` for package **svg-extract**. The macro `\svg@set@input@path` is called to support commands like `\input{`⟨*tex filename*⟩`}` within SVG files.

```
\newsavebox\svg@box
\newcommand*\svg@input{\svg@@input}
\newcommand*\svg@@input[2][]{%
  \IfArgIsEmpty{#1}{}{\svg@local@param@set{#1}}%
  \svg@set@input@path%
```

If the export with ***Inkscape*** was done with LaTeX support enabled, the corresponding file will be used together with `\input`. The necessary patches to environment `picture` as well as command `\includegraphics` are made beforehand with `\svg@patches`.

```
  \@svg@tempswatrue%
  \if@svg@draft%
```

```
    \@svg@tempswafalse%
  \fi%
  \if@svg@ink@latex\else%
    \@svg@tempswafalse%
  \fi%
  \edef\svg@tempa{#2}%
  \if@svg@tempswa%
    \svg@patches{\svg@tempa}%
    \ifnum\value{svg@param@lastpage}=\z@\relax%
      \expandafter\svg@get@lastpage\expandafter{\svg@tempa}%
    \fi%
    \edef\svg@tempa{%
      \ifx\svg@param@pretex\relax\else%
        \noexpand\svg@param@pretex%
      \fi%
      \noexpand\input{\svg@tempa_tex}%
      \ifx\svg@param@apptex\relax\else%
        \noexpand\svg@param@apptex%
      \fi%
    }%
```

If `distort=true` is desired, the input is resized with `\resizebox*`.

```
    \if@svg@param@distort%
      \def\svg@tempb{\resizebox*{\svg@param@width}{\svg@param@height}}%
    \else%
      \let\svg@tempb\@firstofone%
    \fi%
    \sbox\svg@box{\svg@tempb{\svg@tempa}}%
```

If a rotation angle was given, the input is done within `\rotatebox`.

```
    \ifdim\dimexpr\svg@param@angle\p@\relax=\z@\relax%
      \let\svg@tempb\@firstofone%
    \else%
      \edef\svg@tempb{%
        \noexpand\rotatebox[origin=\svg@param@origin]{\svg@param@angle}%
      }%
    \fi%
    \svg@tempb{\usebox\svg@box}%
  \else%
```

If the export with ***Inkscape*** was done without LaTeX support, the resulting graphic file will be included with `\includegraphics`.

```
    \svg@wrn@scale%
    \edef\svg@tempb{%
      draft\if@svg@draft\else=false\fi,%
      scale=\svg@param@scale,%
      keepaspectratio\if@svg@param@distort=false\fi%
    }%
    \ifdim\svg@param@height>\z@\relax%
      \edef\svg@tempb{\svg@tempb,height=\svg@param@height}%
    \fi%
    \ifdim\svg@param@width>\z@\relax%
      \edef\svg@tempb{\svg@tempb,width=\svg@param@width}%
    \fi%
    \ifdim\dimexpr\svg@param@angle\p@\relax=\z@\relax\else%
      \edef\svg@tempb{%
        \svg@tempb,origin=\svg@param@origin,angle=\svg@param@angle%
      }%
    \fi%
    \expandafter\includegraphics\expandafter[\svg@tempb]{\svg@tempa}%
  \fi%
}
```

`\svg@wrn@scale` The option `scale` respectively the parameter `scale` is only considered if the size was not specified.

```
\newcommand*\svg@wrn@scale{%
  \ifdim\dimexpr\svg@param@scale\p@\relax=\p@\relax\else%
    \@svg@tempswafalse%
    \ifdim\svg@param@width>\z@\relax%
      \@svg@tempswatrue%
    \fi%
    \ifdim\svg@param@height>\z@\relax%
      \@svg@tempswatrue%
    \fi%
    \if@svg@tempswa%
      \PackageWarning{svg}{%
        The parameter ‘scale’ is only considered if neither\MessageBreak%
        ‘width’ nor ‘height’ are specified%
      }%
    \fi%
  \fi%
}
```

`\svg@get@lastpage` This macro is used to circumvent the multiple pages bug for PDF files of ***Inkscape*** 0.91, when the the LaTeX export was enabled. For this purpose, the total page number is read from the PDF file.

```
\newcommand*\svg@get@lastpage[1]{%
  \Ifstr{\svg@ink@format}{pdf}{%
    \begingroup%
      \@tempcnta=\m@ne\relax%
      \ifx\XeTeXpdfpagecount\@undefined%
        \ifpdf%
          \ifx\pdfximage\@undefined%
            \ifx\saveimageresource\@undefined\else%
              \saveimageresource{#1}%
              \@tempcnta=\lastsavedimageresourcepages\relax%
            \fi%
          \else%
            \pdfximage{#1}%
            \@tempcnta=\pdflastximagepages\relax%
          \fi%
        \fi%
      \else%
        \@tempcnta=\XeTeXpdfpagecount#1\relax%
      \fi%
      \ifnum\@tempcnta=\m@ne\relax%
        \PackageWarning{svg}{%
          It wasn’t possible to detect the last page\MessageBreak%
          of ‘#1’%
        }%
      \else%
        \PackageInfo{svg}{Last page of ‘#1’ is \the\@tempcnta}%
      \fi%
      \edef\svg@tempa{%
        \endgroup%
        \noexpand\FamilyOptions{SVG}{lastpage=\the\@tempcnta}%
      }%
    \svg@tempa%
  }{}%
}
```

`\svg@file@missing` The error message, which is raised, if a file is missing either after the export with ***Inkscape*** or in general.

```
\newcommand*\svg@file@missing[3][]{%
  \begingroup%
  \svg@quotes@remove[{#2}]{\svg@tempa}%
  \svg@filename@parse[{#1}]{\svg@tempa}%
  \IfArgIsEmpty{#1}{%
   \svg@quotes@remove[{#3}]{\svg@tempb}%
   \def\svg@tempa{%
    Did you run the export with Inkscape? There’s no file\MessageBreak%
    ‘\filename@area\filename@base.\filename@ext’\MessageBreak%
    although ‘\svg@tempb’ was found.%
   }%
  }{%
   \edef\filename@ext{#1}%
   \Ifstr{\filename@area}{./}{\let\filename@area\@empty}{}%
```

Collecting all considered path for the error message.

```
   \edef\svg@tempb{#3}%
   \Ifstr{\svg@tempb}{./}{\let\svg@tempb\@empty}{}%
   \ifx\svg@tempb\@empty%
    \svg@set@input@path%
   \else%
    \svg@set@input@path[\svg@tempb]%
   \fi%
   \ifx\input@path\@undefined%
    \def\svg@tempb{No additional path was given.}%
   \else%
    \def\svg@tempb{Following folders have additionally been searched:}%
    \expandafter\@tfor\expandafter\svg@tempa\expandafter:\expandafter=%
       \input@path\do{%
     \edef\svg@tempb{\svg@tempb\noexpand\MessageBreak\svg@tempa}%
    }%
   \fi%
```

The error message itself.

```
   \def\svg@tempa{%
    There’s no file ‘\filename@base.\filename@ext’\MessageBreak%
    \ifx\filename@area\@empty%
     neither in the current directory nor any other searched\MessageBreak%
     path given by \string\svgpath\space or \string\graphicspath.%
     \MessageBreak\svg@tempb%
    \else%
     in folder ‘\filename@area’.%
    \fi%
   }%
  }%
  \PackageError{svg}{%
   File ‘\filename@base.\filename@ext’ is missing%
  }{\svg@tempa}%
 \endgroup%
}
```

`\svg@ink@ver@settings`
`\svg@ink@ver@detect`
`\svg@@ink@ver@detect`
`\if@svg@ink@ver@detect`

As the command line interface of ***Inkscape*** has changed between versions `0.x` and `1.x`, option `inkscapeversion=detect` allows to detect the used version of ***Inkscape*** in order to define the calling macro `\svg@ink@cmd`. The obtained version is stored in `\svg@ink@ver`, whereas the following meanings are applied:

`-1` version check has not be done or ***Inkscape*** could not be found/executed

`0` ***Inkscape*** version `0.x` was found

`1` ***Inkscape*** version `1.x` or later was found

All necessary information are stored within `\svg@ink@ver@settings` as three arguments, whereas the first one is the manually set version, the second is the used `inkscapeexe` for automatic detection and the third one is the detected version itself.

```
\newcommand*\svg@ink@ver@settings{{\svg@ink@ver}{\svg@ink@exe}{\m@ne}}
```

```
\newif\if@svg@ink@ver@detect
```

In order to run the check for ***Inkscape*** version at the beginning of the document only if needed, changes of both `inkscapeversion`—at this point stored in `\svg@ink@ver`— as well as `inkscapeexe` are detected and are triggering the version check. After evaluating the triggers, the current values set are stored as two tokens in `\svg@ink@ver@settings`. If a check has been triggered, the detected version will be evaluated further on and is stored in the third token of `\svg@ink@ver@settings`.

```
\newcommand*\svg@ink@ver@detect[3]{%
  \@svg@ink@ver@detectfalse%
  \ifnum\pdf@shellescape=\@ne\relax%
    \ifnum\svg@ink@ver=\m@ne\relax%
```

If `inkscapeexe` was not changed…

```
      \svg@sanitize@dq\svg@tempa{#2}%
      \ifx\svg@tempa\svg@ink@exe%
```

…then enforce the check after a change of mode to `detect`…

```
        \ifnum#1>\m@ne\relax%
          \@svg@ink@ver@detecttrue%
```

…or if detection was never invoked, do so.

```
        \else%
          \ifnum#3=\m@ne\relax%
            \@svg@ink@ver@detecttrue%
          \fi%
        \fi%
```

Enforce the check after a change of `inkscapeexe`.

```
      \else%
        \@svg@ink@ver@detecttrue%
      \fi%
    \fi%
  \fi%
```

After evaluating the last settings and maybe setting the trigger for version detection, the current settings are stored in the main aux file. The detected version will be expanded during the write to the aux file.

```
  \edef\svg@ink@ver@settings{%
    {\svg@ink@ver}{\svg@ink@exe}{\noexpand\svg@ink@ver}%
  }%
```

Run detection if necessary and store the result in `\svg@ink@ver`…

```
  \if@svg@ink@ver@detect%
    \svg@@ink@ver@detect%
  \else%
```

…or otherwise set previous detected version in automatic mode.

```
    \ifnum\svg@ink@ver=\m@ne\relax%
      \def\svg@ink@ver{#3}%
    \fi%
  \fi%
}
```

If the switch `\if@svg@ink@ver@detect` was set by `\svg@ink@ver@detect` during the evaluation of `\svg@ink@settings`, which holds the settings of the last run, this macro is invoked. The call of ***Inkscape*** stored in `\svg@ink@exe` is done with `\@@input|"’...’ -V"` in order to read from `stdout`.

```
\newcommand*\svg@@ink@ver@detect{%
  \begingroup%
    \@makeother\|%
    \@makeother\&%
    \everyeof{\noexpand}%
    \svg@quotes@remove{\svg@ink@exe}%
```

Keep in mind, that ***Inkscape*** uses `stderr` [sic] to report the used version of Pango, but as MiKTEX has been fixed by now, **svg** does not have to consider anymore.

```
    \edef\svg@tempa{%
      \edef\noexpand\svg@tempa{\noexpand\@@input|"’\svg@ink@exe’ -V " }%
    }%
    \svg@tempa%
    \trim@spaces@in\svg@tempa%
```

The found version is stored in `\svg@tempa` and parsed afterwards.

```
    \long\def\svg@tempb ##1Inkscape ##2.##3\@nil{%
      \gdef\svg@ink@ver{##2}%
```

If no version was detected, a warning is written.

```
      \ifnum\svg@ink@ver=\m@ne\relax%
        \PackageWarning{svg}{%
          No version of Inkscape was detected by invoking\MessageBreak%
          ‘\svg@ink@exe\space-V’\MessageBreak%
          so the Inkscape export will fail quite sure as the\MessageBreak%
          command is probably unknown to your OS. You could set\MessageBreak%
          ‘inkscapeversion=<version>’ manually but this is very\MessageBreak%
          unlikely to work%
        }%
      \fi%
    }%
    \expandafter\svg@tempb\svg@tempa Inkscape \m@ne.\@nil%
  \endgroup%
}
```

Comparing the stored settings from last the last run with current settings.

```
\AtBeginDocument{\expandafter\svg@ink@ver@detect\svg@ink@ver@settings}
```

Writing `\svg@ink@exe` and `\svg@ink@ver` to the main aux-file.

```
\BeforeClosingMainAux{%
  \if@filesw%
    \immediate\write\@mainaux{%
      \string\gdef\string\svg@ink@ver@settings{\svg@ink@ver@settings}%
    }%
  \fi%
}
```

`\svg@ink@ver@explore` If detecting the used version automatically with `inkscape -V` was not successful, it is tried to explore this by calling ***Inkscape*** through its command line interface with all known variations. If the desired file was created the used version is stored in `\svg@ink@ver`.

```
\newcommand*\svg@ink@ver@explore[3]{%
  \begingroup%
    \@svg@tempswafalse%
    \@tfor\svg@ink@ver:={1}{0}\do{%
      \ShellEscape{\svg@ink@cmd{#1}{#2}}%
      \IfFileExists{#3.\svg@ink@format}{\@svg@tempswatrue}{}%
```

An output file was found, break loop.

```
      \if@svg@tempswa%
        \@break@tfor%
      \fi%
    }%
```

If even this attempt fails, an error message is shown.

```
    \if@svg@tempswa%
      \xdef\svg@ink@ver{\svg@ink@ver}%
    \else%
      \PackageError{svg}{Inkscape version not detected}{%
        It was tried to invoke ‘\svg@ink@exe’\MessageBreak%
        for file "#1.\svg@file@ext"\MessageBreak%
        but no result was produced. Check the log file\MessageBreak%
        and set ‘inkscapeversion=<version>’ manually.%
      }%
    \fi%
  \endgroup%
}
```

B.4. Handling path and file names

`\svg@set@input@path`
`\svg@append@input@path`

In order to import SVG files from different folders, `\svg@set@input@path` evaluates several macros, which are supposed to be used for holding different search folders. Any given path will be handled by `\svg@normalize@path`. The optional argument can be used to append an additional search path.

```
\newcommand*\svg@set@input@path[1][]{%
  \begingroup%
    \svg@deactivate@dq%
```

If a path was already found and stored within `\svg@file@path`, it is searched first and wrapped in curly braces. This is necessary for using commands like `\input{`⟨*tex filename*⟩`}` within SVG files.

```
    \ifx\svg@file@path\@empty\else%
      \svg@normalize@path{\svg@file@path}%
      \edef\svg@file@path{{\svg@file@path}}%
    \fi%
```

Afterwards, several search paths are appended. If `\svgpath` was used, it is searched next. If nothing was found, `\graphicspath` is considered if defined followed by a path given in the third argument. If nothing was found yet, the standard `\input@path` is searched last.

```
    \svg@append@input@path{\svg@file@path}{\svg@input@path}%
    \svg@append@input@path{\svg@file@path}{\Ginput@path}%
    \IfArgIsEmpty{#1}{}{\svg@append@input@path{\svg@file@path}{{#1}}}%
    \svg@append@input@path{\svg@file@path}{\input@path}%
```

Finally, `\input@path` is set.

```
    \edef\svg@tempa{%
      \endgroup%
      \ifx\svg@file@path\@empty\else%
        \def\noexpand\input@path{\svg@file@path}%
      \fi%
    }%
  \svg@tempa%
}
```

Only, if a certain search path is defined, it is added. The paths given in the first argument are compared to each path in the second argument and only new ones are added.

```
\newcommand*\svg@append@input@path[2]{%
  \ifx#2\@undefined\else%
    \edef\svg@tempb{#2}%
    \expandafter\@tfor\expandafter\svg@tempa\expandafter:\expandafter=%
        \svg@tempb\do{%
```

Passing each new path to `\svg@normalize@path`. If a path already exists, switch `\if@svg@tempswa` is set to `false`.

```
      \ifx\svg@tempa\@empty\else%
        \@svg@tempswatrue%
        \svg@normalize@path{\svg@tempa}%
        \expandafter\@tfor\expandafter\svg@tempb\expandafter:\expandafter=%
            #1\do{%
          \ifx\svg@tempa\svg@tempb%
            \@svg@tempswafalse%
            \@break@tfor%
          \fi%
        }%
        \if@svg@tempswa%
          \edef#1{#1{\svg@tempa}}%
        \fi%
      \fi%
    }%
  \fi%
}
```

`\svg@get@path`
`\if@svg@file@found`
`\svg@file@path`
`\svg@file@name`
`\svg@file@base`
`\svg@file@suffix`

The command `\svg@get@path` tries to find a given SVG file. If the searched file wasn't found in the current path, all paths given with `\svgpath` are evaluated. If there was no appropriate file again, all paths given by `\graphicspath` are examined. In the last step, a given path within the second mandatory argument is browsed. The results for file path and name are stored in `\svg@file@path` and `\svg@file@name` as well as the compound of both is saved in `\svg@file@base`.

```
\newif\if@svg@file@found
\newcommand*\svg@file@path{}
\newcommand*\svg@file@name{}
\newcommand*\svg@file@base{}
\newcommand*\svg@file@suffix{}
\newcommand*\svg@get@path[3][\svg@file@ext]{%
  \begingroup%
    \svg@filename@parse[{#1}]{#2}%
    \IfArgIsEmpty{#1}{%
      \edef\svg@tempa{\filename@area\filename@base.\filename@ext}%
    }{%
      \edef\svg@tempa{\filename@area\filename@base.#1}%
    }%
```

After calling `\svg@set@input@path`, all search paths are stored in `\input@path`, a single path given in the third argument will also be considered.

```
    \svg@set@input@path[{#3}]%
```

The specified file is searched with `\IfFileExists`. If the file search was successful, the macro `\svg@filename@parse` is called with the result.

```
    \@svg@tempswafalse%
    \expandafter\IfFileExists\expandafter{\svg@tempa}{%
      \expandafter\svg@quotes@check\expandafter{\svg@tempa}%
      \if@svg@quotes@found\else%
        \svg@quotes@remove{\@filef@und}%
      \fi%
    \@svg@tempswatrue%
```

```
      \edef\@filef@und{\expandafter\trim@spaces\expandafter{\@filef@und}}%
      \svg@filename@parse[{#1}]{\@filef@und}%
    }{}%
    \edef\svg@tempa{%
      \endgroup%
      \if@svg@tempswa%
        \noexpand\@svg@file@foundtrue%
        \def\noexpand\svg@file@path{\filename@area}%
        \def\noexpand\svg@file@name{\filename@base}%
        \def\noexpand\svg@file@base{\filename@area\filename@base}%
      \else%
        \noexpand\@svg@file@foundfalse%
        \def\noexpand\svg@file@path{}%
        \def\noexpand\svg@file@name{#2}%
        \def\noexpand\svg@file@base{#2}%
      \fi%
    }%
  \svg@tempa%
}
```

B.5. Patches

\svg@patches
\svg@picture@saved
\svg@includegraphics@saved

For including the export results from ***Inkscape*** with LaTeX support enabled, there are some patches necessary for environment `picture` and `\includegraphics`. These patches are done with `\svg@patches`.

```
\newcommand*\svg@picture@saved{}
\let\svg@picture@saved\picture
\newcommand*\svg@includegraphics@saved{}
\let\svg@includegraphics@saved\includegraphics
\newcommand*\svg@patches[1]{%
  \let\picture\svg@picture@patched%
  \let\includegraphics\svg@includegraphics@patched%
  \edef\svg@includegraphics@file{#1}%
}
```

\svg@picture@patched
\svg@pictur@patched

In order to provide the possibility specify the desired width of a graphic, the appropriate `\unitlength` is calculated at the beginning of the `picture` environment.

```
\newcommand*\svg@picture@patched{}
\newcommand*\svg@pictur@patched{}
\long\def\svg@picture@patched#1{\svg@pictur@patched@#1}
\def\svg@pictur@patched@(#1,#2){%
  \svg@wrn@scale%
```

If a desired height is present, the resulting `\unitlength` is calculated with the ratio of the coordinates of the `picture` environment given as arguments for x- and y-direction by using `\Gscale@div`. With this factor, `\unitlength`—which is connected to the x-coordinate—can be scaled in a suitable manner.

```
  \ifdim\svg@param@height>\z@\relax%
    \Gscale@div\svg@tempa{#1\p@}{#2\p@}%
    \setlength\unitlength{\svg@param@height}%
    \setlength\unitlength{\svg@tempa\unitlength}%
    \ifdim\svg@param@width>\z@\relax%
      \ifdim\unitlength>\svg@param@width\relax%
        \setlength\unitlength{\svg@param@width}%
      \fi%
    \fi%
  \else%
```

If no height is given, `\unitlength` can be set easily.

```
    \ifdim\svg@param@width>\z@\relax%
```

```
      \setlength\unitlength{\svg@param@width}%
    \else%
      \setlength\unitlength{\svg@param@scale\unitlength}%
    \fi%
  \fi%
```

After setting `\unitlength`, the `picture` environment can be called with its original definition.

```
  \svg@picture@saved(#1,#2)%
}
```

`\svg@includegraphics@patched`
`\svg@includegraphics@file`

The patch to `\includegraphics` is meant to dissolve the ***Inkscape*** bug concerning the inclusion of more PDF pages than actually are existing.

The given optional parameters to `\includegraphics` are processed and the counter `svg@param@currpage` is set to the value of a given `page`. The value of parameter `width` is ignored.

```
\DefineFamily{SVGpatch}
\DefineFamilyMember{SVGpatch}
\newcounter{svg@param@currpage}
\setcounter{svg@param@currpage}{\m@ne}
\FamilyCounterKey{SVGpatch}{page}{svg@param@currpage}
\DefineFamilyKey{SVGpatch}{width}{\FamilyKeyStateProcessed}
\newcommand*\svg@includegraphics@file{}
\newcommand*\svg@includegraphics@patched[2][]{%
  \FamilyOptions{SVGpatch}{#1}%
```

If option `lastpage` was set to `false`, each page is included—even if it doesn't exist, which may cause errors.

```
  \ifnum\value{svg@param@lastpage}<\z@\relax%
    \FamilySetCounter{SVGpatch}{page}{svg@param@currpage}{%
      \the\value{svg@param@lastpage}%
    }%
  \fi%
```

Only if counter `svg@param@lastpage` is smaller than `svg@param@currpage`, pages are included, where `svg@param@lastpage` was either given as a number with parameter `lastpage` or was automatically calculated with `\svg@get@lastpage`.

```
  \ifnum\value{svg@param@currpage}>\value{svg@param@lastpage}\relax\else%
```

A page is included with the original definition of `\includegraphics`. All optional parameters are passed.

```
    \svg@includegraphics@saved[{#1}]{\svg@includegraphics@file}%
  \fi%
}
```

C. Extracting independent graphic files with svg-extract

C.1. Options

For package **svg-extract** the user-interface of package **svg** is extended. The following options can either be set with `\svgsetup` or be used as optional parameters for `\includesvg` and `\includeinkscape`.

`\svg@dummy@key` If package **svg-extract** wasn't loaded, the following options are defined for package **svg** in order to raise a warning message. Primarily this is done for compatibility reasons.

```
⟨*main⟩
\DefineFamilyMember[.dummy]{SVG}
\newcommand*\svg@dummy@key[2][]{%
  \@ifpackageloaded{svg-extract}{}{%
    \IfArgIsEmpty{#1}{%
      \DefineFamilyKey[.dummy]{SVG}{#2}{%
        \PackageWarning{svg}{%
          The option key ‘#2’ can only\MessageBreak%
          be used with package ‘svg-extract’, but\MessageBreak%
          you didn’t load it%
        }%
        \FamilyKeyStateProcessed%
      }%
    }{%
      \DefineFamilyKey[.dummy]{SVG}{#2}[{#1}]{%
        \PackageWarning{svg}{%
          The option key ‘#2’ can only\MessageBreak%
          be used with package ‘svg-extract’, but\MessageBreak%
          you didn’t load it%
        }%
        \FamilyKeyStateProcessed%
      }%
    }%
```

Before package **svg-extract** the given key `#2` of family member `.dummy` is relaxed.

```
    \AfterPackage{svg-extract}{\RelaxFamilyKey[.dummy]{SVG}{#2}}%
  }%
}
⟨/main⟩
```

C.1.1. Controlling the extract process

`extract` (opt.)
`\if@svgx@run`

With option `extract` it can be controlled, if the extraction of independent graphic files should be done.

```
⟨*main⟩
\svg@dummy@key[true]{extract}
⟨/main⟩
⟨*extract⟩
\newif\if@svgx@run
\DefineFamilyKey{SVG}{extract}[true]{%
  \lowercase{\def\svg@tempa{#1}}%
  \FamilySetNumerical{SVG}{extract}{svg@tempa}{%
    {false}{0},{off}{0},{no}{0},%
    {true}{1},{on}{1},{yes}{1},{onlynewer}{1},{newer}{1},%
    {overwrite}{1},{force}{1},{forced}{1},%
    {pdf}{2},{eps}{3},{ps}{4}%
  }{\svg@tempa}%
  \ifx\FamilyKeyState\FamilyKeyStateProcessed%
    \ifcase\svg@tempa\relax% false
      \@svgx@runfalse%
    \or% true
      \@svgx@runtrue%
    \or% pdf
      \FamilyOptions{SVG}{extractformat=pdf}%
    \or% eps
      \FamilyOptions{SVG}{extractformat=eps}%
    \or% ps
      \FamilyOptions{SVG}{extractformat=ps}%
    \fi%
  \fi%
}
⟨/extract⟩
```

`on` (opt.)
`off` (opt.)

Package options which can be used to switch functionality on or off during the loading of package **svg-extract**.

```
⟨*extract⟩
\DeclareOption{on}{\FamilyOptions{SVG}{extract=true}}
\DeclareOption{off}{\FamilyOptions{SVG}{extract=false}}
⟨/extract⟩
```

`extractformat` (opt.)
`\svgx@format`
`pdf` (opt.)
`eps` (opt.)

Option `extractformat` controls the output format (`pdf/eps/ps`). It is set to `pdf` or, if dvi output could be detected, to `eps` during initialization.

```
⟨*main⟩
\svg@dummy@key{extractformat}
\svg@dummy@key[true]{pdf}
\svg@dummy@key[true]{eps}
⟨/main⟩
⟨*extract⟩
\newcommand*\svgx@format{pdf}
\ifxetex\else\ifpdf\else
  \renewcommand*\svgx@format{eps}
\fi\fi
\DefineFamilyKey{SVG}{extractformat}{%
  \lowercase{\edef\svgx@format{#1}}%
  \FamilyKeyStateProcessed%
}
\DefineFamilyKey{SVG}{pdf}[true]{%
  \FamilySetBool{SVG}{pdf}{@svg@tempswa}{#1}%
  \ifx\FamilyKeyState\FamilyKeyStateProcessed%
    \if@svg@tempswa%
      \svgx@ifinlist{pdf}{\svgx@format}{}{%
        \edef\svgx@format{\svgx@format,pdf}%
      }%
      \svg@deprecated@key{pdf}{extractformat={\svgx@format}}%
    \else%
      \FamilyKeyStateUnknownValue%
    \fi%
  \fi%
}
\DefineFamilyKey{SVG}{eps}[true]{%
  \FamilySetBool{SVG}{eps}{@svg@tempswa}{#1}%
  \ifx\FamilyKeyState\FamilyKeyStateProcessed%
    \if@svg@tempswa%
      \svgx@ifinlist{eps}{\svgx@format}{}{%
        \edef\svgx@format{\svgx@format,eps}%
      }%
      \svg@deprecated@key{eps}{extractformat={\svgx@format}}%
    \else%
      \FamilyKeyStateUnknownValue%
    \fi%
  \fi%
}
⟨/extract⟩
```

`extractpreamble` (opt.)
`preamble` (opt.)
`\svgx@preamble`
`extractpreambleend` (opt.)
`end` (opt.)
`\svgx@endpreamble`

For the extraction process, a preamble is necessary for a separate auxiliary LaTeX file. By default, the preamble of the main document is used, which end is detected at `\begin{document}`.

```
⟨*main⟩
\svg@dummy@key{extractpreamble}
\svg@dummy@key{preamble}
\svg@dummy@key{extractpreambleend}
\svg@dummy@key{end}
```

```
⟨/main⟩
⟨*extract⟩
\newcommand*\svgx@preamble{\jobname.\svgx@latex@ext}%
\DefineFamilyKey{SVG}{extractpreamble}{%
  \renewcommand*\svgx@preamble{#1}%
  \FamilyKeyStateProcessed%
}
\DefineFamilyKey{SVG}{preamble}{%
  \svg@deprecated@key[svg-extract]{preamble=#1}{extractpreamble=#1}%
}
\newcommand*\svgx@endpreamble{}
\expandafter\def\expandafter\svgx@endpreamble\expandafter{%
  \csname begin\endcsname{document}%
}
\DefineFamilyKey{SVG}{extractpreambleend}{%
  \renewcommand*\svgx@endpreamble{#1}%
  \FamilyKeyStateProcessed%
}
\DefineFamilyKey{SVG}{end}{%
  \svg@deprecated@key[svg-extract]{end=#1}{extractpreambleend=#1}%
}
⟨/extract⟩
```

`extractruns` (opt.)
`svgx@runs` (counter)

With this option, the number of LaTeX runs for the separate auxiliary file can be set.

```
⟨*main⟩
\svg@dummy@key{extractruns}
⟨/main⟩
⟨*extract⟩
\newcounter{svgx@runs}
\DefineFamilyKey{SVG}{extractruns}{%
  \FamilySetCounter{SVG}{extractruns}{svgx@runs}{#1}%
  \ifx\FamilyKeyState\FamilyKeyStateProcessed%
    \ifnum\value{svgx@runs}<\@ne\relax%
      \PackageWarning{svg-extract}{%
        The count for runs has to be at least one%
      }%
      \FamilySetCounter{SVG}{extractruns}{svgx@runs}{\@ne}%
    \fi%
  \fi%
}
⟨/extract⟩
```

`latexexe` (opt.)
`pdflatex` (opt.)
`\svgx@latex@exe`
`latexext` (opt.)
`\svgx@latex@ext`
`latexopt` (opt.)
`\svgx@latex@opt`

The command and facultative options for the LaTeX call of the separate auxiliary file. The default is set according to the currently used engine.

```
⟨*main⟩
\svg@dummy@key{latexexe}
\svg@dummy@key{pdflatex}
\svg@dummy@key{latexext}
\svg@dummy@key{latexopt}
⟨/main⟩
⟨*extract⟩
\ifxetex
  \newcommand*\svgx@latex@exe{xelatex}
\else\ifluatex
  \ifpdf
    \newcommand*\svgx@latex@exe{lualatex}
  \else
    \newcommand*\svgx@latex@exe{lualatex --output-format=dvi}
  \fi
\else\ifpdf
  \newcommand*\svgx@latex@exe{pdflatex}
\else
  \newcommand*\svgx@latex@exe{latex}
```

```
\fi\fi\fi
\DefineFamilyKey{SVG}{latexexe}{%
  \renewcommand*\svgx@latex@exe{#1}%
  \FamilyKeyStateProcessed%
}
\DefineFamilyKey{SVG}{pdflatex}{%
  \svg@deprecated@key[svg-extract]{pdflatex=#1}{latexexe=#1}%
}
\newcommand*\svgx@latex@ext{tex}
\DefineFamilyKey{SVG}{latexext}{%
  \renewcommand*\svgx@latex@ext{#1}%
  \FamilyKeyStateProcessed%
}
\newcommand*\svgx@latex@opt{}
\DefineFamilyKey{SVG}{latexopt}{%
  \renewcommand*\svgx@latex@opt{#1}%
  \FamilyKeyStateProcessed%
}
⟨/extract⟩
```

`dvipsopt` (opt.)
`\svgx@dvips@exe`
`\svgx@dvips@opt`
`pstoepsopt` (opt.)
`\svgx@pstoeps@exe`
`\svgx@pstoeps@opt`
`pstopdfopt` (opt.)
`\svgx@pstopdf@exe`
`\svgx@pstopdf@opt`
`pdftoepsopt` (opt.)
`\svgx@pdftoeps@exe`
`\svgx@pdftoeps@opt`
`pdftopsopt` (opt.)
`\svgx@pdftops@exe`
`\svgx@pdftops@opt`
`pdftops` (opt.)

Options and macros for calling convert commands, which are supplied by most LaTeX distributions. These are used to generate all files, which are supported by option `extractformat`, as they don't need an additional application.

```
⟨*main⟩
\svg@dummy@key{dvipsopt}
\svg@dummy@key{pstoepsopt}
\svg@dummy@key{pstopdfopt}
\svg@dummy@key{pdftoepsopt}
\svg@dummy@key{pdftopsopt}
\svg@dummy@key{pdftops}
⟨/main⟩
⟨*extract⟩
\newcommand*\svgx@dvips@exe{dvips}
\newcommand*\svgx@dvips@opt{}
\DefineFamilyKey{SVG}{dvipsopt}{%
  \renewcommand*\svgx@dvips@opt{#1}%
  \FamilyKeyStateProcessed%
}
\newcommand*\svgx@pstoeps@exe{ps2eps}
\newcommand*\svgx@pstoeps@opt{-B -C}
\DefineFamilyKey{SVG}{pstoepsopt}{%
  \renewcommand*\svgx@pstoeps@opt{#1}%
  \FamilyKeyStateProcessed%
}
\newcommand*\svgx@pstopdf@exe{ps2pdf}
\newcommand*\svgx@pstopdf@opt{}
\DefineFamilyKey{SVG}{pstopdfopt}{%
  \renewcommand*\svgx@pstopdf@opt{#1}%
  \FamilyKeyStateProcessed%
}
\newcommand*\svgx@pdftoeps@exe{pdftops -eps}
\newcommand*\svgx@pdftoeps@opt{}
\DefineFamilyKey{SVG}{pdftoepsopt}{%
  \renewcommand*\svgx@pdftoeps@opt{#1}%
  \FamilyKeyStateProcessed%
}
\newcommand*\svgx@pdftops@exe{pdftops}
\newcommand*\svgx@pdftops@opt{}
\DefineFamilyKey{SVG}{pdftopsopt}{%
  \renewcommand*\svgx@pdftops@opt{#1}%
  \FamilyKeyStateProcessed%
}
\DefineFamilyKey{SVG}{pdftops}{%
  \PackageWarning{svg-extract}{%
```

```
    The option key ‘pdftops’ is deprecated. \MessageBreak%
    You should use either ‘pdftoepsopt’ or\MessageBreak%
    ‘pdftopsopt’ instead. See the manual for\MessageBreak%
    more. Nothing was done%
  }%
  \FamilyKeyStateProcessed%
}
⟨/extract⟩
```

C.1.2. Invoking external application for graphic conversion

Besides the use of a conversion tool supplied by the LaTeX distribution, the applications ***ImageMagick*** and ***Ghostscript*** can be used for converting graphics.

`convert` (opt.)
`\if@svgx@cnv@run`
`\svgx@cnv@cmd`

The option `convert` can be used to define, which of both applications should be use. ***ImageMagick*** is set by default.

```
⟨*main⟩
\svg@dummy@key[true]{convert}
⟨/main⟩
⟨*extract⟩
\newif\if@svgx@cnv@run
\newcommand*\svgx@cnv@cmd{}
\DefineFamilyKey{SVG}{convert}[true]{%
  \FamilySetNumerical{SVG}{convert}{svg@tempa}{%
    {false}{0},{off}{0},{no}{0},%
    {true}{1},{on}{1},{yes}{1},{onlynewer}{1},{newer}{1},%
    {overwrite}{1},{force}{1},{forced}{1},%
    {magick}{2},{imagemagick}{2},{convert}{2},%
    {gs}{3},{ghostscript}{3},%
    {gs64}{4},{ghostscript64}{4},%
    {gs32}{5},{ghostscript32}{5}%
  }{#1}%
  \ifx\FamilyKeyState\FamilyKeyStateProcessed%
    \ifcase\svg@tempa\relax% false
      \@svgx@cnv@runfalse%
    \or% true
      \@svgx@cnv@runtrue%
    \or% magick
      \@svgx@cnv@runtrue%
      \renewcommand*\svgx@cnv@cmd{\svgx@magick@cmd}%
    \or% gs
      \@svgx@cnv@runtrue%
      \renewcommand*\svgx@cnv@cmd{\svgx@gs@cmd}%
    \or% gs64
      \@svgx@cnv@runtrue%
      \renewcommand*\svgx@cnv@cmd{\svgx@gs@cmd}%
      \svg@ifwindowsdetected{%
        \renewcommand*\svgx@gs@exe{gswin64c}%
      }{}%
    \or% gs32
      \@svgx@cnv@runtrue%
      \renewcommand*\svgx@cnv@cmd{\svgx@gs@cmd}%
      \svg@ifwindowsdetected{%
        \renewcommand*\svgx@gs@exe{gswin32c}%
      }{}%
    \fi%
```

In version v1.0 the option `convert` was used to set both the executable and options for the conversion application, meant for the usage of ***ImageMagick***. This is taken into account here.

```
  \else% legacy option
```

Same doing like with legacy part of option `inkscape`.

```
    \def\svg@tempa##1-##2\@nil{%
      \IfArgIsEmpty{##2}{\def\svg@tempb{}}{%
        \def\svg@tempa##1####1\@nil{\def\svg@tempb{####1}}%
        \svg@tempa#1\@nil%
      }%
      \def\svg@tempa{##1}%
    }%
    \svg@tempa#1-\@nil%
    \PackageWarning{svg-extract}{%
      Setting the executable%
      \ifx\svg@tempb\@empty\else%
        \space and associated options%
      \fi%
      \MessageBreak%
      for ImageMagick should be done with options\MessageBreak%
      ‘magickexe=\svg@tempa’%
      \ifx\svg@tempb\@empty\else%
        \MessageBreak and ‘magicksetting’ and/or ‘magickoperator’%
      \fi.\MessageBreak%
      Nevertheless, this was done by now%
      \ifx\svg@tempb\@empty\else%
        , whereby \MessageBreak ‘magicksetting=\svg@tempb’ was used%
      \fi%
    }%
    \FamilyOptions{SVG}{convert=magick}%
    \edef\svg@tempa{%
      \noexpand\FamilyOptions{SVG}{magickexe=\svg@tempa}%
      \ifx\svg@tempb\@empty\else%
        \noexpand\FamilyOptions{SVG}{magicksetting=\svg@tempb}%
      \fi%
    }%
    \svg@tempa%
  \fi%
}
⟨/extract⟩
```

convertformat (opt.)
\svgx@cnv@format
png (opt.)

Option `convertformat` controls the output format for converted files. It is set to `png` by default.

```
⟨*main⟩
\svg@dummy@key{convertformat}
\svg@dummy@key[true]{png}
⟨/main⟩
⟨*extract⟩
\newcommand*\svgx@cnv@format{png}
\DefineFamilyKey{SVG}{convertformat}{%
  \lowercase{\edef\svgx@cnv@format{#1}}%
  \ifx\svgx@cnv@format\@empty\else%
    \@svgx@cnv@runtrue%
  \fi%
  \FamilyKeyStateProcessed%
}
\DefineFamilyKey{SVG}{png}[true]{%
  \FamilySetBool{SVG}{png}{@svg@tempswa}{#1}%
  \ifx\FamilyKeyState\FamilyKeyStateProcessed%
    \if@svg@tempswa%
      \svgx@ifinlist{png}{\svgx@cnv@format}{}{%
        \edef\svgx@cnv@format{\svgx@cnv@format,png}%
      }%
      \svg@deprecated@key{png}{convertformat={\svgx@cnv@format}}%
    \else%
      \FamilyKeyStateUnknownValue%
    \fi%
  \fi%
}
⟨/extract⟩
```

`convertdpi` (opt.)
`convertdensity` (opt.)
`\svgx@cnv@dpi`
`\svgx@cnv@get@dpi`

The option `convertdpi` is meant to define the used density during the conversion process. It can be set either for all designated output formats or targeted for a specific format. It's also possible to use something like `500x300`. Given values are resolved by `\svgx@cnv@get@dpi`. It's used like `convertdpi=300` or `convertdpi={png=600}` If the option is used for a specific or for all output formats is recognized by `\svgx@ifkeyandval`.

```
⟨*main⟩
\svg@dummy@key{convertdpi}
\svg@dummy@key{convertdensity}
⟨/main⟩
⟨*extract⟩
\newcommand*\svgx@cnv@dpi{}
\let\svgx@cnv@dpi\relax
\DefineFamilyKey{SVG}{convertdpi}{%
  \FamilyKeyStateUnknownValue%
  \svgx@ifkeyandval{#1}{%
    \svgx@cnv@get@dpi{##2}%
    \ifx\svg@tempa\relax\else%
      \expandafter\edef\csname svgx@cnv@dpi@##1\endcsname{\svg@tempa}%
      \FamilyKeyStateProcessed%
    \fi%
  }{%
    \svgx@cnv@get@dpi{##1}%
    \ifx\svg@tempa\relax\else%
      \edef\svgx@cnv@dpi{\svg@tempa}%
      \FamilyKeyStateProcessed%
    \fi%
  }%
}
\DefineFamilyKey{SVG}{convertdensity}{\FamilyOptions{SVG}{convertdpi=#1}}
```

This macro is used to resolve a given value to set the density for the conversion. The delimited macros `\svg@tempa` and `\svg@tempb` are defined to first crop any given suffix `dpi` and second to split two numbers at `x`, if present. Pay attention how both macros are invoked. In the end, a passed value in any of the forms `300`, `300dpi`, `300x400` or `300x400dpi` and even `300dpix400dpi` is possible. The result is stored in `\svg@tempa`.

```
\newcommand*\svgx@cnv@get@dpi[1]{%
  \begingroup%
    \def\svg@tempa##1dpi##2x##3dpi##4\@nil{%
      \edef\svg@tempa{##1}%
```

Switch `\if@svg@tempswa` as `\iftrue` means, a valid value was found.

```
      \@svg@tempswafalse%
```

If only the first argument is a number and third is empty, a single number was given and there's nothing more to do. If the argument is something like `300dpix400dpi`, the third argument is the second number.

```
      \Ifnumber{##1}{%
        \IfArgIsEmpty{##3}{\@svg@tempswatrue}{%
          \Ifnumber{##3}{\edef\svg@tempa{##1x##3}}{}%
        }%
      }{}%
      \if@svg@tempswa\else%
        \expandafter\svg@tempb\svg@tempa xx\@nil%
      \fi%
    }%
```

Macro `\svg@tempb` splits at `x` and checks, if something valid like `300x400` was given. If true, the value is stored in `\svg@tempa`.

```
    \def\svg@tempb##1x##2x##3\@nil{%
      \Ifstr{##3}{x}{%
        \@svg@tempswatrue%
        \IfArgIsEmpty{##1}{\@svg@tempswafalse}{%
          \Ifnumber{##1}{}{\@svg@tempswafalse}%
        }%
        \IfArgIsEmpty{##2}{\@svg@tempswafalse}{%
          \Ifnumber{##2}{}{\@svg@tempswafalse}%
        }%
        \if@svg@tempswa%
          \edef\svg@tempa{##1x##2}%
        \fi%
      }{}%
    }%
    \IfArgIsEmpty{#1}{%
      \let\svg@tempa\@empty%
    }{%
      \lowercase{\svg@tempa#1dpi#1xdpi\@nil}%
      \if@svg@tempswa\else%
        \let\svg@tempa\relax%
      \fi%
    }%
    \edef\svg@tempb{%
      \endgroup%
      \ifx\svg@tempa\relax%
        \let\noexpand\svg@tempa\noexpand\relax%
      \else%
        \def\noexpand\svg@tempa{\svg@tempa}%
      \fi%
    }%
  \svg@tempb%
}
⟨/extract⟩
```

`\svgx@setformatkey`
`\svgx@useformatkey`

With `\svgx@setformatkey` the—maybe output format depend—keys for the conversion tools are set. First argument contains the value given to a key, second the command sequence name of the macro, to whom the value shall be allocated.

```
\newcommand*\svgx@setformatkey[2]{%
```

A key of the form `⟨key⟩={⟨format⟩=⟨value⟩}` is given. The desired output format can be accessed with `##1`, the value with `##2` within the arguments of `\svgx@ifkeyandval`.

```
  \svgx@ifkeyandval{#1}{%
    \svg@ifvalueisrelax{##2}{%
      \expandafter\let\csname #2@##1\endcsname\relax%
    }{%
      \@namedef{#2@##1}{##2}%
    }%
```

A key of the form `⟨key⟩={⟨format⟩=⟨value⟩}` is given. The value can be used with `##1`.

```
  }{%
    \svg@ifvalueisrelax{##1}{%
      \expandafter\let\csname #2\endcsname\relax%
    }{%
      \@namedef{#2}{##1}%
    }%
  }%
}
```

The command `\svgx@useformatkey` checks, if a format specific key was defined with `\svgx@setformatkey`, whereas the format is given in the second argument. If this is not the case, the setting for all output formats is used. After that, a specific key appended with a `+` can be used to do some additional stuff.

```
\newcommand*\svgx@useformatkey[3]{%
  \scr@ifundefinedorrelax{#1@#2}{%
    \scr@ifundefinedorrelax{#1}{}{%
      \expandafter\ifx\csname #1\endcsname\@empty\else%
        #3\@nameuse{#1}\space%
      \fi%
    }%
    \scr@ifundefinedorrelax{#1@#2+}{}{%
      \expandafter\ifx\csname #1@#2+\endcsname\@empty\else%
        #3\@nameuse{#1@#2+}\space%
      \fi%
    }%
  }{%
```

If a format specific key was definded, it is used.

```
    \expandafter\ifx\csname #1@#2\endcsname\@empty\else%
      #3\@nameuse{#1@#2}\space%
    \fi%
  }%
}
```

`magickexe` (opt.)
`\svgx@magick@exe`
`magicksetting` (opt.)
`\svgx@magick@set`
`magickoperator` (opt.)
`\svgx@magick@opr`
`\svgx@magick@cmd`

Setting the command including maybe the path to ***ImageMagick***. The keys `magicksetting` and `magickoperator` should be used to add optional arguments before (*Settings*) or after (*Operators*) the input file. They can either be set for all or a specific output format as like option `convertdpi`. For this `\svgx@setformatkey` is used.

```
⟨*main⟩
\svg@dummy@key{magickexe}
\svg@dummy@key{magicksetting}
\svg@dummy@key{magickoperator}
⟨/main⟩
⟨*extract⟩
\svg@ifwindowsdetected{%
  \newcommand*\svgx@magick@exe{magick}%
}{%
  \newcommand*\svgx@magick@exe{convert}%
}
\DefineFamilyKey{SVG}{magickexe}{%
  \renewcommand*\svgx@magick@exe{#1}%
  \FamilyKeyStateProcessed%
}
\newcommand*\svgx@magick@set{}
\DefineFamilyKey{SVG}{magicksetting}{%
  \svgx@setformatkey{#1}{svgx@magick@set}%
  \FamilyKeyStateProcessed%
}
\newcommand*\svgx@magick@opr{}
\DefineFamilyKey{SVG}{magickoperator}{%
  \svgx@setformatkey{#1}{svgx@magick@opr}%
  \FamilyKeyStateProcessed%
}
\newcommand*\svgx@magick@cmd[3]{%
  \svgx@magick@exe\space%
  \svgx@useformatkey{svgx@cnv@dpi}{#3}{-density }%
  \svgx@useformatkey{svgx@magick@set}{#3}{}%
  "#1.#2"\space%
  \svgx@useformatkey{svgx@magick@opr}{#3}{}%
  "#1.#3"%
}
⟨/extract⟩
```

`gsexe` (opt.)
`\svgx@gs@exe`
`gsopt` (opt.)
`\svgx@gs@opt`
`gsdevice` (opt.)
`\svgx@gs@device`
`\svgx@gs@cmd`

Options to set the command including maybe the path to ***Ghostscript***. As ***Ghostscript*** needs a specific device defined for different output formats, the option `gsdevice` can be used. It can either be set for all or a specific output format just like `gsopt` in the same manner like option `convertdpi`.

```
⟨*main⟩
\svg@dummy@key{gsexe}
\svg@dummy@key{gsopt}
\svg@dummy@key{gsdevice}
⟨/main⟩
⟨*extract⟩
\svg@ifwindowsdetected{%
  \newcommand*\svgx@gs@exe{gswin64c}%
}{%
  \newcommand*\svgx@gs@exe{gs}%
}
\DefineFamilyKey{SVG}{gsexe}{%
  \renewcommand*\svgx@gs@exe{#1}%
  \FamilyKeyStateProcessed%
}
\newcommand*\svgx@gs@opt{}
\DefineFamilyKey{SVG}{gsopt}{%
  \svgx@setformatkey{#1}{svgx@gs@opt}%
  \FamilyKeyStateProcessed%
}
\newcommand*\svgx@gs@device{}
\DefineFamilyKey{SVG}{gsdevice}{%
  \svgx@setformatkey{#1}{svgx@gs@device}%
  \FamilyKeyStateProcessed%
}
\newcommand*\svgx@gs@cmd[3]{%
  \svgx@gs@exe\space-dSAFER -dBATCH -dNOPAUSE\space%
  \svgx@useformatkey{svgx@gs@device}{#3}{-sDEVICE=}%
  \svgx@useformatkey{svgx@cnv@dpi}{#3}{-r}%
  \svgx@useformatkey{svgx@gs@opt}{#3}{}%
  -sOutputFile="#1.#3"\space"#1.#2"%
}
⟨/extract⟩
```

C.1.3. Setting output folder

`extractpath` (opt.)
`path` (opt.)
`\svgx@out@path`

The option `extractpath` controls, in which folder the results both of the extraction as well as the conversion of ***ImageMagick*** or ***Ghostscript*** will be located.

```
⟨*main⟩
\svg@dummy@key{extractpath}
\svg@dummy@key{path}
⟨/main⟩
⟨*extract⟩
\newcommand*\svgx@out@path{}
\DefineFamilyKey{SVG}{extractpath}{%
  \svg@sanitize@dq\svg@tempb{#1}%
  \FamilySetNumerical{SVG}{extractpath}{svg@tempa}{%
    {svgpath}{0},{svgdir}{0},%
    {svgsubpath}{1},{svgsubdir}{1},%
    {basepath}{2},{basedir}{2},{jobpath}{2},{jobdir}{2},%
    {basesubpath}{3},{basesubdir}{3},{jobsubpath}{3},{jobsubdir}{3}%
  }{\svg@tempb}%
  \ifx\FamilyKeyState\FamilyKeyStateProcessed%
    \ifcase\svg@tempa\relax% svgpath
      \renewcommand*\svgx@out@path{\svg@file@path}%
    \or% svgsubpath
      \renewcommand*\svgx@out@path{\svg@file@path svg-extract/}%
    \or% basepath
```

```
      \renewcommand*\svgx@out@path{./}%
    \or% basesubpath
      \renewcommand*\svgx@out@path{./svg-extract/}%
    \fi%
  \else%
    \edef\svgx@out@path{\svg@tempb}%
    \svg@normalize@path{\svgx@out@path}%
    \FamilyKeyStateProcessed%
  \fi%
}
\DefineFamilyKey{SVG}{path}{%
  \svg@deprecated@key[svg-extract]{path=#1}{extractpath=#1}%
}
⟨/extract⟩
```

`extractname` (opt.)
`name` (opt.)
`\svgx@out@name`
`\if@svgx@out@sec`
`svgx@out@count` (counter)
`\svgx@out@sec`

With option `extractname` the name of the extracted and maybe converted file itself can be changed.

```
⟨*main⟩
\svg@dummy@key{extractname}
\svg@dummy@key{name}
⟨/main⟩
⟨*extract⟩
\newcommand*\svgx@out@name{}
\newif\if@svgx@out@sec
\newcounter{svgx@out@count}
\newcommand*\svgx@out@sec{unknown}
\DefineFamilyKey{SVG}{extractname}{%
  \svg@quotes@remove[{#1}]{\svg@tempb}%
  \FamilySetNumerical{SVG}{extractname}{svg@tempa}{%
    {filename}{0},{name}{0},%
    {filenamenumbered}{1},{namenumbered}{1},%
    {numberedfilename}{1},{numberedname}{1},%
    {numbered}{2},{section}{2},{numberedsection}{2},{sectionnumbered}{2}%
  }{\svg@tempb}%
  \@svgx@out@secfalse%
  \ifx\FamilyKeyState\FamilyKeyStateProcessed%
    \ifcase\svg@tempa\relax% filename
      \renewcommand*\svgx@out@name{\svg@out@name-extract}%
    \or% filenamenumbered
      \renewcommand*\svgx@out@name{\the\value{svgx@out@count}-\svg@out@name}%
    \or% numbered
      \renewcommand*\svgx@out@name{\the\value{svgx@out@count}-\svgx@out@sec}%
      \@svgx@out@sectrue%
    \fi%
  \else%
    \if@svg@quotes@found%
      \edef\svgx@out@name{"\svg@tempb"}%
    \else%
      \edef\svgx@out@name{\svg@tempb}%
    \fi%
    \FamilyKeyStateProcessed%
  \fi%
}
\DefineFamilyKey{SVG}{name}{%
  \svg@deprecated@key[svg-extract]{name=#1}{extractname=#1}%
}
⟨/extract⟩
```

C.1.4. Options for the extraction of graphics

`extractwidth` (opt.)
`\svgx@param@width`
`extractheight` (opt.)
`\svgx@param@width`
`extractdistort` (opt.)
`extractkeepaspectratio` (opt.)
`\svgx@param@distort`
`extractscale` (opt.)
`\svgx@param@scale`

For graphic extraction, the given settings regarding the size for inclusion can be overwritten with these options. Using `\relax` as value leads to resetting an option as unset, regardless of what was previously given. The value `inherit` means, that the actual option for including is used for extraction as well. This is the default setting.

```
⟨*main⟩
\svg@dummy@key{extractwidth}
\svg@dummy@key{extractheight}
\svg@dummy@key{extractdistort}
\svg@dummy@key{extractkeepaspectratio}
\svg@dummy@key{extractscale}
⟨/main⟩
⟨*extract⟩
\newcommand*\svgx@param@width{\svg@param@width}
\DefineFamilyKey{SVG}{extractwidth}{%
  \FamilyKeyStateUnknownValue%
  \svg@ifvalueisrelax{#1}{%
    \renewcommand*\svgx@param@width{\z@}%
    \FamilyKeyStateProcessed%
  }{%
    \Ifstr{#1}{inherit}{%
      \renewcommand*\svgx@param@width{\svg@param@width}%
      \FamilyKeyStateProcessed%
    }{%
      \FamilySetLengthMacro{SVG}{extractwidth}{\svgx@param@width}{#1}%
      \ifx\FamilyKeyState\FamilyKeyStateProcessed%
        \ifdim\svgx@param@width<\z@\relax%
          \FamilyKeyStateUnknownValue%
        \fi%
      \fi%
    }%
  }%
}
\newcommand*\svgx@param@height{\svg@param@height}
\DefineFamilyKey{SVG}{extractheight}{%
  \FamilyKeyStateUnknownValue%
  \svg@ifvalueisrelax{#1}{%
    \renewcommand*\svgx@param@height{\z@}%
    \FamilyKeyStateProcessed%
  }{%
    \Ifstr{#1}{inherit}{%
      \renewcommand*\svgx@param@height{\svg@param@height}%
      \FamilyKeyStateProcessed%
    }{%
      \FamilySetLengthMacro{SVG}{extractheight}{\svgx@param@height}{#1}%
      \ifx\FamilyKeyState\FamilyKeyStateProcessed%
        \ifdim\svgx@param@height<\z@\relax%
          \FamilyKeyStateUnknownValue%
        \fi%
      \fi%
    }%
  }%
}
\newif\if@svgx@param@distort
\DefineFamilyKey{SVG}{extractdistort}[true]{%
  \FamilyKeyStateUnknownValue%
  \svg@ifvalueisrelax{#1}{%
    \@svgx@param@distortfalse%
    \FamilyKeyStateProcessed%
  }{%
    \Ifstr{#1}{inherit}{%
      \renewcommand*\if@svgx@param@distort{\if@svg@param@distort}%
      \FamilyKeyStateProcessed%
    }{%
      \FamilySetBool{SVG}{extractdistort}{@svgx@param@distort}{#1}%
    }%
  }%
}
```

```
\DefineFamilyKey{SVG}{extractkeepaspectratio}[true]{%
  \FamilySetBool{SVG}{extractkeepaspectratio}{@svg@tempswa}{#1}%
  \ifx\FamilyKeyState\FamilyKeyStateProcessed%
    \if@svg@tempswa%
      \FamilyOptions{SVG}{extractdistort=false}%
    \else%
      \FamilyOptions{SVG}{extractdistort=true}%
    \fi%
  \else%
    \FamilyOptions{SVG}{extractdistort=#1}%
  \fi%
}
\newcommand*\svgx@param@scale{\svg@param@scale}
\DefineFamilyKey{SVG}{extractscale}{%
  \FamilyKeyStateUnknownValue%
  \svg@ifvalueisrelax{#1}{%
    \renewcommand*\svgx@param@scale{1}%
    \FamilyKeyStateProcessed%
  }{%
    \Ifstr{#1}{inherit}{%
      \renewcommand*\svgx@param@scale{\svg@param@scale}%
      \FamilyKeyStateProcessed%
    }{%
      \Ifisdimension{#1\p@}{%
        \ifdim\dimexpr#1\p@\relax>\z@\relax%
          \renewcommand*\svgx@param@scale{#1}%
          \FamilyKeyStateProcessed%
        \fi%
      }{}%
    }%
  }%
}
⟨/extract⟩
```

`extractpretex` (opt.)
`\svgx@param@pretex`
`extractapptex` (opt.)
`\svgx@param@apptex`
`extractpostex` (opt.)

The similar hooks for executing code right before or after the graphic extraction.

```
⟨*main⟩
\svg@dummy@key{extractpretex}
\svg@dummy@key{extractapptex}
\svg@dummy@key{extractpostex}
⟨/main⟩
⟨*extract⟩
\newcommand*\svgx@param@pretex{\svg@param@pretex}
\DefineFamilyKey{SVG}{extractpretex}{%
  \svg@ifvalueisrelax{#1}{%
    \let\svgx@param@pretex\relax%
  }{%
    \Ifstr{#1}{inherit}{%
      \renewcommand*\svgx@param@pretex{\svg@param@pretex}%
    }{%
      \renewcommand*\svgx@param@pretex{#1}%
    }%
  }%
  \FamilyKeyStateProcessed%
}
\newcommand*\svgx@param@apptex{\svg@param@apptex}
\DefineFamilyKey{SVG}{extractapptex}{%
  \svg@ifvalueisrelax{#1}{%
    \let\svgx@param@apptex\relax%
  }{%
    \Ifstr{#1}{inherit}{%
      \renewcommand*\svgx@param@apptex{\svg@param@apptex}%
    }{%
      \renewcommand*\svgx@param@apptex{#1}%
    }%
  }%
  \FamilyKeyStateProcessed%
}
\DefineFamilyKey{SVG}{extractpostex}{%
  \svg@deprecated@key[svg-extract]{extractpostex=#1}{extractapptex=#1}%
}
⟨/extract⟩
```

C.1.5. Miscellaneous options

`clean` (opt.)
`clear` (opt.)
`\svgx@clean`

With option `clean` files generated during the extraction process can be deleted. Setting `true` will remove all files, `false` won't clear any file. Additionally, a specific file list of suffixes can be given.

```
⟨*main⟩
\svg@dummy@key[true]{clean}
\svg@dummy@key[true]{clear}
⟨/main⟩
⟨*extract⟩
\newcommand*\svgx@clean{}
\DefineFamilyKey{SVG}{clean}[true]{%
  \FamilySetBool{SVG}{clean}{@svg@tempswa}{#1}%
  \ifx\FamilyKeyState\FamilyKeyStateProcessed%
    \if@svg@tempswa%
      \renewcommand*\svgx@clean{log,aux,dvi,out,ps,eps,pdf,\svgx@latex@ext}%
    \else%
      \renewcommand*\svgx@clean{}%
    \fi%
  \else%
    \renewcommand*\svgx@clean{#1}%
    \FamilyKeyStateProcessed%
  \fi%
}
\DefineFamilyKey{SVG}{clear}[true]{\FamilyOptions{SVG}{clean=#1}}
⟨/extract⟩
```

`exclude` (opt.)

If it is desired not to include but only extract graphics with package **svg-extract**, option `exclude` can be used.

```
⟨*main⟩
\svg@dummy@key[true]{exclude}
⟨/main⟩
⟨*extract⟩
\DefineFamilyKey{SVG}{exclude}[true]{%
  \FamilySetBool{SVG}{exclude}{@svg@tempswa}{#1}%
  \ifx\FamilyKeyState\FamilyKeyStateProcessed%
    \if@svg@tempswa%
      \renewcommand*\svg@input[2][]{%
        \if@svgx@run\else%
          \PackageWarning{svg-extract}{%
            The image '##2' was\MessageBreak%
            neither extracted nor included%
          }%
        \fi%
      }%
    \else%
      \renewcommand*\svg@input{\svg@@input}%
    \fi%
  \fi%
}
⟨/extract⟩
```

C.2. User commands

\includesvg
extract (param.)
extractpreamble (param.)
extractformat (param.)
extractwidth (param.)
extractheight (param.)
extractdistort (param.)
extractscale (param.)
extractangle (param.)
extractpretex (param.)
extractapptex (param.)
extractruns (param.)
latexexe (param.)
latexopt (param.)
latexext (param.)
dvipsopt (param.)
pstoepsopt (param.)
pstopdfopt (param.)
pdftoepsopt (param.)
pdftopsopt (param.)
convert (param.)
convertformat (param.)
convertdpi (param.)
magicksetting (param.)
magickoperator (param.)
gsopt (param.)
gsdevice (param.)
clean (param.)
exclude (param.)

The parameters `angle` and `origin` are defined as pendants to the keys provided by `\includegraphics`.

```
⟨*extract⟩
\newcommand*\svgx@param@angle{0}
\svg@local@param@def{%
  \DefineFamilyKey[.param]{SVG}{extractangle}{%
    \FamilyKeyStateUnknownValue%
    \svg@ifvalueisrelax{#1}{%
      \renewcommand*\svgx@param@angle{0}%
      \FamilyKeyStateProcessed%
    }{%
      \Ifstr{#1}{inherit}{%
        \renewcommand*\svgx@param@angle{\svg@param@angle}%
        \FamilyKeyStateProcessed%
      }{%
        \Ifisdimension{#1\p@}{%
          \renewcommand*\svgx@param@angle{#1}%
          \FamilyKeyStateProcessed%
        }{}%
      }%
    }%
  }%
}
⟨/extract⟩
```

\svghidepreamblestart
\svghidepreambleend

Some dummys for package **svg**.

```
⟨*main⟩
\newcommand*\svghidepreamblestart{%
  \PackageWarning{svg}{%
    The macro ‘\string\svghidepreamblestart’ is only meant\MessageBreak%
    to be used together with package ‘svg-extract’. \MessageBreak%
    Nevertheless, nothing will happen%
  }%
}
\newcommand*\svghidepreambleend{%
  \PackageWarning{svg}{%
    The macro ‘\string\svghidepreambleend’ is only meant\MessageBreak%
    to be used together with package ‘svg-extract’. \MessageBreak%
    Nevertheless, nothing will happen%
  }%
}
⟨/main⟩
```

\includeinkscape
extract (param.)
extractpreamble (param.)
extractformat (param.)
extractwidth (param.)
extractheight (param.)
extractdistort (param.)
extractscale (param.)
extractangle (param.)
extractpretex (param.)
extractapptex (param.)
extractruns (param.)
latexexe (param.)
latexopt (param.)
latexext (param.)
dvipsopt (param.)
pstoepsopt (param.)
pstopdfopt (param.)
pdftoepsopt (param.)
pdftopsopt (param.)
convert (param.)
convertformat (param.)
convertdpi (param.)
magicksetting (param.)
magickoperator (param.)
gsopt (param.)
gsdevice (param.)
clean (param.)
exclude (param.)

These two macros can be used to hide some parts of the preamble during reading the preamble of the main document.

```
⟨*extract⟩
\let\svghidepreamblestart\relax
\let\svghidepreambleend\relax
⟨/extract⟩
```

C.3. Auxiliary macros

\svg@extract

The macro `\svg@extract` does the actual job of both extracting and converting independent graphic files. Since it is necessary to run it with `--shell-escape` enabled, the command raises a warning if it is not activated. Afterwards, the package is finished.

```
⟨*main⟩
\newcommand*\svg@extract[1]{}
⟨/main⟩
```

```
⟨*extract⟩
\ifnum\pdf@shellescape=\@ne\relax\else%
  \renewcommand*\svg@extract[1]{%
    \if@svgx@run%
      \begingroup%
        \edef\svg@tempa{#1}%
        \svg@quotes@remove{\svg@tempa}%
        \PackageWarning{svg-extract}{%
          You didn’t enable ‘shell escape’ (or ‘write18’)\MessageBreak%
          so it wasn’t possible to run the extraction for\MessageBreak%
          file ‘\svg@tempa’\MessageBreak%
        }%
      \endgroup%
    \fi%
  }%
  \expandafter\endinput%
\fi
⟨/extract⟩
```

`\svgx@stream@in`
`\svgx@read@line`
`\svgx@stream@out`
`\if@svgx@preamble@write`

Both an input stream and an output stream are necessary for this as well as a token register, which is used to store all commands which should be executed on shell.

```
\newread\svgx@stream@in
\newcommand*\svgx@read@line{}
\newwrite\svgx@stream@out
\newif\if@svgx@preamble@write
```

`\svg@extract`

If flag `--shell-escape` is enabled, the command is defined with its intended functionality. It runs all necessary processes to extract and convert graphic files.

```
\renewcommand*\svg@extract[1]{%
```

If option `extract` is enabled…

```
  \if@svgx@run%
```

…the macro `\svgx@get@out@sec` is used to get the current level numbering within the document and the counter for extracted graphics is stepped. After that, a separate auxiliary LaTeX file is created for extracting independent graphic files. The macro `\svgx@get@out@sec` is used to get the current level numbering within the document. The specified preamble is read for this task, if it exists. It is first searched in the same folder as the SVG file and if it wasn’t found, in any other valid folder for SVG files.

```
    \if@svgx@out@sec%
      \svgx@get@out@sec%
    \fi%
    \stepcounter{svgx@out@count}%
    \begingroup%
      \def\svg@tempa##1.##2\@nil{%
        \IfArgIsEmpty{##2}{\edef\svgx@preamble{##1.\svgx@latex@ext}}{}%
      }%
      \expandafter\svg@tempa\svgx@preamble.\@nil%
      \IfFileExists{\svg@file@path\svgx@preamble}{%
        \@svg@file@foundtrue%
      }{%
        \svg@get@path[]{\svgx@preamble}{\svg@out@path}%
        \def\svg@tempa####1.####2\@nil{%
          \edef\svgx@preamble{\svg@file@name.####2}%
        }%
        \expandafter\svg@tempa\svgx@preamble\@nil%
      }%
      \edef\svg@tempa{%
        \endgroup%
        \if@svg@file@found%
          \ifx\svg@file@path\@empty%
        \def\noexpand\svgx@preamble{./\svgx@preamble}%
      \else%
        \def\noexpand\svgx@preamble{\svg@file@path\svgx@preamble}%
      \fi%
    \fi%
  }%
\svg@tempa%
\begingroup%
  \endlinechar=\m@ne%
  \IfFileExists{\svgx@preamble}{%
    \PackageInfo{svg-extract}{%
      The preamble file `\svgx@preamble'\MessageBreak%
      is used for the generation of the auxiliary file\MessageBreak%
      `\svgx@out@name.\svgx@latex@ext'%
    }%
```

The catcodes for `#` need to be changed to prevent doubling when reading the line.

```
    \catcode`\#=12\relax%
    \immediate\openout\svgx@stream@out=\svgx@out@name.\svgx@latex@ext%
    \immediate\openin\svgx@stream@in=\svgx@preamble%
    \@svg@tempswatrue%
    \@svgx@preamble@writetrue%
    \def\svgx@read@line{}%
```

The given preamble file is read line by line and written to the separate auxiliary LaTeX file `\svgx@out@name.\svgx@latex@ext` via the output stream.

```
    \@whilesw\if@svg@tempswa\fi{%
      \immediate\read\svgx@stream@in to\svgx@read@line%
      \ifx\svgx@read@line\@empty%
        \ifeof\svgx@stream@in\@svg@tempswafalse\fi%
      \else%
```

With `\svghidepreamblestart` and `\svghidepreambleend` it is possible for the user to omit certain parts of the preamble. Therefor the two macros `\svgx@read@preamble@till` and `\svgx@read@preamble@from` are toggling the switch `\if@svgx@preamble@write`

```
        \svgx@read@preamble@till{\svghidepreamblestart}{}%
        \svgx@read@preamble@from{\svghidepreambleend}{}%
```

If the desired end of the preamble (`\svgx@endpreamble`) was found, the readout is terminated by switching `\if@svg@tempswa` to `false`.

```
        \svgx@read@preamble@till{\svgx@endpreamble}{\@svg@tempswafalse}%
        \if@svgx@preamble@write%
```

During the readout process, it is searched with `\svgx@documentclass` for the appearance of `\documentclass` and `\if@svgx@classfound` is set to `true` if it was found.

```
          \if@svgx@classfound\else%
            \expandafter\svgx@documentclass%
              \svgx@read@line\documentclass\documentclass\@nil%
          \fi%
```

Writing out the—maybe manipulated—read in line.

```
          \ifx\svgx@read@line\@empty\else%
            \immediate\write\svgx@stream@out{%
              \unexpanded\expandafter{\svgx@read@line}%
            }%
          \fi%
        \fi%
      \fi%
    }%
    \immediate\closein\svgx@stream@in%
    \immediate\closeout\svgx@stream@out%
    \catcode`\#=6\relax%
```

Once the separate auxiliary LaTeX file is written, it is read in again and its content is stored in `\svg@tempa`, since it is necessary to prepend some stuff to the preamble, for example a maybe not existent document class.

```
     \immediate\openin\svgx@stream@in=\svgx@out@name.\svgx@latex@ext%
     \def\svg@tempa{}%
     \loop\unless\ifeof\svgx@stream@in%
      \readline\svgx@stream@in to\svgx@read@line%
      \ifx\svgx@read@line\@empty\else%
       \edef\svg@tempa{%
        \unexpanded\expandafter{\svg@tempa}%
        \unexpanded\expandafter{\svgx@read@line}^^J%
       }%
      \fi%
     \repeat%
     \immediate\closein\svgx@stream@in%
    }{%
```

If a file was given that doesn’t exist, a warning is issued.

```
     \svg@quotes@remove{\svgx@preamble}%
     \ifx\svgx@preamble\@empty\else%
      \PackageWarning{svg-extract}{%
       The preamble file ‘\svgx@preamble’\MessageBreak%
       does not exist%
      }%
     \fi%
     \def\svg@tempa{}%
    }%
```

After the preamble was read in and stored in `\svg@tempa`, the separate auxiliary LaTeX file is written again. Some information are written right at the beginning of the file.

```
    \immediate\openout\svgx@stream@out=\svgx@out@name.\svgx@latex@ext%
    \immediate\write\svgx@stream@out{%
     \@percentchar\@percentchar\space This file was generated by package
     ‘svg-extract’^^J%
     \@percentchar\@percentchar\space from source ‘\jobname’^^J%
     \@percentchar\@percentchar\space It’s intended to be compiled with
     ‘\svgx@latex@exe\ifx\svgx@latex@opt\@empty\else\space\svgx@latex@opt\fi’
    }%
```

With the intention of passing the correct paper dimensions, the calculating of the paper size is executed with `\AtBeginDocument` even before the document class, so that this is definitely the first thing to happen at the beginning of the document. Additionally, it is ensured that the `\special` command is definitely used with the correct paper size, when creating a DVI file.

```
    \immediate\write\svgx@stream@out{%
     \string\AtBeginDocument{\@percentchar^^J%
      \space\space\string\svgxsetpapersize\@percentchar^^J%
      \ifxetex\else\ifpdf\else%
       \space\space\string\AtBeginDvi{\string\special{%
         papersize=\string\the\string\paperwidth,%
          \string\the\string\paperheight%
       }}\@percentchar^^J%
      \fi\fi%
     }^^J%
     \string\PassOptionsToPackage{hidelinks}{hyperref}%
    }%
```

If no document class was found during reading the preamble file, then class `\article` is used.

```
    \if@svgx@classfound\else%
     \immediate\write\svgx@stream@out{\string\documentclass{article}}%
    \fi%
```

And now the stored preamble.

```
    \ifx\svg@tempa\@empty\else%
      \immediate\write\svgx@stream@out{\unexpanded\expandafter{\svg@tempa}}%
    \fi%
```

After the given preamble was written, package **svg-extract** will be loaded in case it was forgotten.

```
    \immediate\write\svgx@stream@out{\string\usepackage{svg-extract}}%
```

Now all parameters relevant for the extraction are evaluated and appended.

```
    \def\svg@tempa##1{%
      \immediate\write\svgx@stream@out{\string\svgsetup{##1}}%
    }%
    \if@svg@ink@latex\else%
      \svg@tempa{inkscapelatex=false}%
    \fi%
    \ifdim\svgx@param@width>\z@\relax%
      \svg@tempa{width=\svgx@param@width}%
    \fi%
    \ifdim\svgx@param@height>\z@\relax%
      \svg@tempa{height=\svgx@param@height}%
    \fi%
    \if@svgx@param@distort%
      \svg@tempa{distort=true}%
    \fi%
    \ifdim\dimexpr\svgx@param@scale\p@\relax=\p@\relax\else%
      \svg@tempa{scale=\svgx@param@scale}%
    \fi%
    \def\svg@tempb{\svg@param@pretex}%
    \ifx\svgx@param@pretex\svg@tempb\relax%
      \let\svgx@param@pretex\svg@param@pretex%
    \fi%
    \ifx\svgx@param@pretex\relax\else%
      \svg@tempa{pretex=\unexpanded\expandafter{\svgx@param@pretex}}%
    \fi%
    \def\svg@tempb{\svg@param@apptex}%
    \ifx\svgx@param@apptex\svg@tempb\relax%
      \let\svgx@param@apptex\svg@param@apptex%
    \fi%
    \ifx\svgx@param@apptex\relax\else%
      \svg@tempa{apptex=\unexpanded\expandafter{\svgx@param@apptex}}%
    \fi%
```

Parameter `lastpage` is only considered for including PDF files with LaTeX support.

```
    \let\svg@tempa\@empty%
    \if@svg@ink@latex%
      \Ifstr{\svg@ink@format}{pdf}{%
        \ifnum\value{svg@param@lastpage}>\z@\relax%
          \edef\svg@tempa{lastpage=\the\value{svg@param@lastpage}}%
        \else%
          \ifnum\value{svg@param@lastpage}=\z@\relax%
            \def\svg@tempa{lastpage=true}%
          \else%
            \def\svg@tempa{lastpage=false}%
          \fi%
        \fi%
      }{}%
    \fi%
```

The rotation angle, if given.

```
    \ifdim\dimexpr\svgx@param@angle\p@\relax=\z@\relax\else%
      \edef\svg@tempa{%
```

```
        angle=\svgx@param@angle\ifx\svg@tempa\@empty\else,\svg@tempa\fi%
      }%
    \fi%
```

As this is now the end of the preamble and just before the beginning of the document, the paper dimension are set again to make sure, that these settings are active at the end of the preamble. Additionally, it is executed again at the very end of `\AtBeginDocument` to ensure, that no other package used this hook for manipulating the paper size.

```
    \ifx\svg@tempa\@empty%
      \def\svg@tempa{\string\svgxsetbox{#1}}%
    \else%
      \edef\svg@tempa{\noexpand\string\noexpand\svgxsetbox[\svg@tempa]{#1}}%
    \fi%
    \immediate\write\svgx@stream@out{\svg@tempa}%
```

Package **xr** is used to evaluate possible labels within the included ***Inkscape*** LaTeX file.

```
    \if@svg@ink@latex%
      \IfFileExists{xr.sty}{%
        \immediate\write\svgx@stream@out{%
          \string\usepackage{xr}^^J%
          \string\externaldocument{\jobname}^^J%
        }%
      }{}%
    \fi%
    \immediate\write\svgx@stream@out{%
      \string\begin{document}^^J%
      \string\pagestyle{empty}^^J%
      \string\svgxoutputbox\@percentchar^^J%
      \string\end{document}%
    }%
    \immediate\closeout\svgx@stream@out%
  \endgroup%
```

After creating the separate auxiliary LaTeX file, the actual extraction and conversion can be done.

```
  \Ifstr{\svgx@format\svgx@cnv@format}{}{%
    \PackageWarning{svg-extract}{%
      Both keys ‘extractformat’ and ‘convertformat’ are\MessageBreak%
      empty, so nothing to do so far%
    }%
  }{%
```

As the extraction maybe needs to include the main auxiliary file with `\externaldocument` provided by package **xr** it is necessary to do all related stuff after the main auxiliary file was written. This is done with `\AfterReadingMainAux` provided by package **scrlfile**.

```
    \svg@quotes@remove{\svgx@out@path}%
    \svg@quotes@remove{\svgx@out@name}%
```

All generated files will be moved to the desired output folder, which is given by option `extractpath`. Therefor, this folder is created.

```
    \edef\svg@tempb{%
      \noexpand\svg@shell@mkdir{\svgx@out@path}%
    }%
    \expandafter\AfterReadingMainAux\expandafter{\svg@tempb}%
```

First of all the separate auxiliary LaTeX file is compiled with the detected LaTeX engine (`\svgx@latex@exe`) as often as defined by counter option `extractruns`.

```
    \edef\svg@tempb{%
      \noexpand\PackageInfo{svg-extract}{%
        Running LaTeX (\svgx@latex@exe) for graphic extraction%
        \ifx\svgx@latex@opt\@empty\else%
```

```
          \MessageBreak with added options ‘\svgx@latex@opt’%
        \fi%
      }%
    }%
    \expandafter\AfterReadingMainAux\expandafter{\svg@tempb}%
    \edef\svg@tempb{%
      \noexpand\ShellEscape{%
        \svgx@latex@exe\space\svgx@latex@opt\space%
        "\svgx@out@name.\svgx@latex@ext"%
      }%
    }%
    \loop\ifnum\value{svgx@runs}>\z@\relax%
      \expandafter\AfterReadingMainAux\expandafter{\svg@tempb}%
      \advance\c@svgx@runs\m@ne%
    \repeat%
```

All files requested with option `extractformat` are created with internal conversion tools supplied by most LaTeX distributions if necessary.

```
    \def\svg@tempa##1##2##3{%
      \edef\svg@tempb{%
        \noexpand\ShellEscape{%
          \@nameuse{svgx@##1@exe}\space\@nameuse{svgx@##1@opt}\space%
          "\svgx@out@name.##2"%
        }%
      }%
      \AfterReadingMainAux{\PackageInfo{svg-extract}{Running ##1}}%
      \expandafter\AfterReadingMainAux\expandafter{\svg@tempb}%
    }%
    \@svg@tempswafalse%
    \ifxetex\else\ifpdf\else%
      \@svg@tempswatrue%
    \fi\fi%
    \if@svg@tempswa%
      \svg@tempa{dvips}{dvi}{ps}%
      \svgx@ifinlist{eps}{\svgx@format}{\svg@tempa{pstoeps}{ps}{eps}}{}%
      \svgx@ifinlist{pdf}{\svgx@format}{\svg@tempa{pstopdf}{ps}{pdf}}{}%
    \else%
      \svgx@ifinlist{eps}{\svgx@format}{\svg@tempa{pdftoeps}{pdf}{eps}}{}%
      \svgx@ifinlist{ps}{\svgx@format}{\svg@tempa{pdftops}{pdf}{ps}}{}%
    \fi%
```

Now the desired conversion tool is invoked if requested.

```
    \if@svgx@cnv@run%
```

If no density was given at all, the density for PNG files is set to `300dpi` by default.

```
      \ifx\svgx@cnv@dpi\relax%
        \ifx\svgx@cnv@dpi@png\@undefined%
          \def\svgx@cnv@dpi@png{300}%
        \fi%
      \fi%
```

The first given file type with option `extractformat` is used as source for the conversion process.

```
      \expandafter\svgx@cnv@get@informat\expandafter{\svgx@format}%
```

The conversion is done for each desired file type given in a list by option `convertformat`.

```
      \@for\svg@tempa:=\svgx@cnv@format\do{%
        \ifx\svg@tempa\@empty\else%
          \expandafter\svgx@ifinlist\expandafter{\svg@tempa}{\svgx@format}{%
            \PackageWarning{svg-extract}{%
              File type ‘\svg@tempa’ was specified for option\MessageBreak%
              ‘extractformat’ (\svgx@format) as well as for \MessageBreak%
```

```
                option ‘convertformat’ (\svgx@cnv@format) so the\MessageBreak%
                conversion won’t be done%
              }%
            }{%
              \edef\svg@tempb{%
                \noexpand\PackageInfo{svg-extract}{%
                  Converting ‘\svgx@out@name.\svgx@cnv@informat’\MessageBreak%
                  to ‘\svgx@out@name.\svg@tempa’%
                }%
              }%
              \expandafter\AfterReadingMainAux\expandafter{\svg@tempb}%
              \edef\svg@tempb{%
                \noexpand\ShellEscape{%
                  \svgx@cnv@cmd{\svgx@out@name}{\svgx@cnv@informat}{\svg@tempa}%
                }%
              }%
              \expandafter\AfterReadingMainAux\expandafter{\svg@tempb}%
            }%
          \fi%
        }%
      \fi%
```

As both extraction and conversion are done, all files are moved to the desired output folder (`extractpath`).

```
      \edef\svg@tempa{\svgx@format\if@svgx@cnv@run,\svgx@cnv@format\fi}%
      \@for\svg@tempb:=\svg@tempa\do{%
        \ifx\svg@tempb\@empty\else%
          \edef\svg@tempb{%
            \noexpand\svgx@move{\svgx@out@name}{\svg@tempb}{\svgx@out@path}%
          }%
          \expandafter\AfterReadingMainAux\expandafter{\svg@tempb}%
        \fi%
      }%
```

At the very end, all unwanted auxiliary files are deleted.

```
      \@for\svg@tempa:=\svgx@clean\do{%
        \expandafter\svgx@ifinlist\expandafter{\svg@tempa}{\svg@tempb}{}{%
          \edef\svg@tempb{%
            \noexpand\IfFileExists{"\svgx@out@name".\svg@tempa}{%
              \noexpand\svg@shell@rm{\svgx@out@name.\svg@tempa}%
            }{}%
          }%
          \expandafter\AtEndDocument\expandafter{%
            \expandafter\AfterReadingMainAux\expandafter{\svg@tempb}%
          }%
        }%
      }%
    }%
  \fi%
}
```

`\svgx@get@out@sec`

The macro `\svgx@get@out@sec` reads all sectioning counters in order to get the numbering of the current sectioning level. The value is stored in `\svgx@out@sec`.

```
\newcommand*\svgx@get@out@sec{%
  \begingroup%
    \def\svg@tempa{}%
    \@for\svg@tempb:={%
      part,chapter,section,subsection,subsubsection,paragraph,subparagraph%
    }\do{%
      \ifx\svg@tempb\@empty\else%
        \scr@ifundefinedorrelax{the\svg@tempb}{}{%
          \ifnum\value{\svg@tempb}>\z@\relax%
            \edef\svg@tempa{\svg@tempb}%
          \fi%
        }%
      \fi%
    }%
    \edef\svg@tempb{%
      \endgroup%
      \ifx\svg@tempa\@empty\else%
        \def\noexpand\svgx@out@sec{\csname the\svg@tempa\endcsname}%
      \fi%
    }%
  \svg@tempb%
}
```

`\svgx@documentclass`
`\if@svgx@classfound`

This delimited macro is used to find the occurrence of `\documentclass` within a read in line. The delimiter `\documentclass` is used twice in order to ignore the possible occurrence of white space or anything else right before `\documentclass`.

```
\newif\if@svgx@classfound
\newcommand*\svgx@documentclass{}
\def\svgx@documentclass#1\documentclass#2\documentclass#3\@nil{%
  \IfArgIsEmpty{#2}{}{\@svgx@classfoundtrue}%
}
```

`\svgx@read@preamble@till`
`\svgx@read@preamble@from`
`\svgx@read@preamble@skip`

These macros are used to skip some parts of a read in preamble file.

```
\newcommand*\svgx@read@preamble@till[2]{%
  \svgx@read@preamble@skip#1\@nil{till}{#2}%
}
\newcommand*\svgx@read@preamble@from[2]{%
  \svgx@read@preamble@skip#1\@nil{from}{#2}%
}
```

In principle, the functionality is the same as for `\svgx@documentclass`.

```
\newcommand*\svgx@read@preamble@skip{}
\def\svgx@read@preamble@skip#1\@nil#2#3{%
```

A given token is used to create the macro `\svg@tempa` delimited by the token itself which is used twice to get any stuff right before or after the occurrence.

```
  \def\svg@tempa##1{%
    \def\svg@tempa####1##1####2##1####3\@nil{%
      \IfArgIsEmpty{####3}{}{%
```

Write everything which was found right before the macro which starts hiding area to the output stream and stop writing with `\if@svgx@preamble@write`.

```
        \Ifstr{#2}{till}{%
          \IfArgIsEmpty{####1}{}{%
            \immediate\write\svgx@stream@out{####1}%
          }%
          \@svgx@preamble@writefalse%
        }{%
```

Write everything which was found right after the macro which ends the hiding area and start writing again with `\if@svgx@preamble@write`.

```
          \Ifstr{#2}{from}{%
            \IfArgIsEmpty{####2}{%
              \def\svgx@read@line{}%
            }{%
              \def\svgx@read@line{####2}%
            }%
            \@svgx@preamble@writetrue%
          }{}%
        }%
```

Additonal stuff which should be done.

```
        #3%
      }%
    }%
  }%
```

Creating the macro `\svg@tempa` delimited by the first argument.

```
  \edef\svg@tempb{\expandafter\detokenize\expandafter{#1}}%
  \expandafter\svg@tempa\expandafter{\svg@tempb}%
```

Calling the created macro.

```
  \edef\svg@tempb{%
    \expandafter\detokenize\expandafter{\svgx@read@line}\svg@tempb\svg@tempb%
  }%
  \expandafter\svg@tempa\svg@tempb\@nil%
}
```

`\svgx@cnv@informat`
`\svgx@cnv@get@informat`

The first list entry from argument (`\svgx@format`) is extracted by `\svgx@cnv@get@informat`.

```
\newcommand*\svgx@cnv@informat{}
\newcommand*\svgx@cnv@get@informat[1]{%
  \begingroup%
    \def\svg@tempa##1,##2\@nil{%
      \def\svg@tempa{##1}%
    }%
    \svg@tempa#1,\@nil%
    \edef\svg@tempa{%
      \endgroup%
      \def\noexpand\svgx@cnv@informat{\svg@tempa}%
    }%
  \svg@tempa%
```

If the first argument (`\svgx@format`) was empty, `\svgx@cnv@informat` is set to the a file type, which is generated anyway.

```
  \ifx\svgx@cnv@informat\@empty%
    \renewcommand*\svgx@cnv@informat{pdf}%
    \ifxetex\else\ifpdf\else%
      \renewcommand*\svgx@cnv@informat{ps}%
    \fi\fi%
  \fi%
}
```

`\svgx@move`

If the file doesn't exist

```
\newcommand*\svgx@move[3]{%
  \begingroup%
    \IfFileExists{"#1".#2}{%
      \svg@shell@mv{#1.#2}{#3#1.#2}%
    }{%
      \edef\svg@tempa{#2}%
      \@svg@tempswafalse%
      \expandafter\svgx@ifinlist\expandafter{\svg@tempa}{\svgx@cnv@format}{%
        \@svg@tempswatrue%
        \def\svg@tempb{conversion}%
      }{%
        \expandafter\svgx@ifinlist\expandafter{\svg@tempa}{pdf,ps,eps}{%
          \@svg@tempswatrue%
          \def\svg@tempb{extraction}%
        }{}%
      }%
      \if@svg@tempswa%
        \edef\svg@tempb{%
```

```
          The graphic file \svg@tempb\space failed\MessageBreak%
          for ‘#1.#2’\MessageBreak%
          Troubleshooting: Please check the log file how\MessageBreak%
          the invocation of the extraction took place and\MessageBreak%
          try to execute it yourself in the terminal%
        }%
      \else%
        \def\svg@tempb{%
          The extraction to format ‘#2’ failed\MessageBreak%
          for ‘#1.#2’\MessageBreak%
          Only file types ‘pdf,ps,eps’ are supported for\MessageBreak%
          key ‘exportformat’%
        }%
      \fi%
      \PackageWarning{svg-extract}{\svg@tempb}%
    }%
  \endgroup%
}
```

C.4. Commands for the separate auxiliary LaTeX-file

For the extraction of independent graphics, an auxiliary LaTeX file is needed. Within this file, the following commands are used to include the desired graphic.

`\svgxsetbox`
`\svgx@setbox`
`\if@svgx@standalone`

Within the preamble of the auxiliary LaTeX file, the desired graphic is used to setup a box, which is used both to define the papersize as well as for the output itself. The macro `\svgx@setbox` is executed twice, the first time in the preamble and the second time at the very end of `\AtBeginDocument` if package **etoolbox** was loaded.

The switch `\if@svgx@standalone` is defined for enabling classes to implement a different behaviour for **svg-extract** in standalone mode. for example, TUD-Script-classes are using this switch.

```
\newif\if@svgx@standalone
\newcommand*\svgxsetbox[2][]{%
  \@svgx@standalonetrue%
  \svgx@setbox{#1}{#2}%
  \scr@ifundefinedorrelax{AtEndPreamble}{%
    \let\svg@tempa\@firstofone%
  }{%
    \def\svg@tempa{\AtEndPreamble}%
  }%
  \svg@tempa{\AtBeginDocument{\svgx@setbox{#1}{#2}}}%
}
\newcommand*\svgx@setbox[2]{%
  \sbox\svg@box{\svg@@input[{#1},draft=false]{#2}}%
  \svgxsetpapersize%
}
```

`\svgxsetpapersize`

This macro sets all well known length macros for defining the paper size as well as the type area to the size of `\svg@box`.

```
\newcommand*\svgxsetpapersize{%
  \setlength\paperwidth{\the\wd\svg@box}%
```

Due to the fact, that the lengths for stock- and mediasizes are maybe set to `\relax`, these macros are checked with `\scr@ifundefinedorrelax`.

```
  \scr@ifundefinedorrelax{stockwidth}{}{%
    \setlength\stockwidth{\paperwidth}%
  }%
  \scr@ifundefinedorrelax{mediawidth}{}{%
    \setlength\mediawidth{\paperwidth}%
  }%
  \setlength\textwidth{\paperwidth}%
  \setlength\paperheight{\the\dimexpr\ht\svg@box+\dp\svg@box\relax}%
  \scr@ifundefinedorrelax{stockheight}{}{%
    \setlength\stockheight{\paperheight}%
  }%
  \scr@ifundefinedorrelax{mediaheight}{}{%
    \setlength\mediaheight{\paperheight}%
  }%
  \setlength\textheight{\paperheight}%
```

Any other length regarding the layout is set to have no influence at all. Hence the document has the same size as the graphic.

```
  \hoffset=-1in%
  \oddsidemargin=\z@%
  \evensidemargin=\z@%
  \voffset=-1in%
  \topmargin=\z@%
  \headheight=\z@%
  \headsep=\z@%
  \topskip=\z@%
  \footskip=\z@%
  \marginparsep=\z@%
  \marginparwidth=\z@%
  \marginparpush=\z@%
}
\@onlypreamble\svgxsetpapersize
```

`\svgxoutputbox`
`\if@svgx@beamer`

With `\svgxoutputbox` the created box is displayed.

```
\newif\if@svgx@beamer
\@ifclassloaded{beamer}{\@svgx@beamertrue}{}%
\newcommand*\svgxoutputbox{%
  \begingroup%
    \setlength\parindent{\z@}%
    \setlength\parskip{\z@}%
    \setlength\parfillskip{\z@}%
    \if@svgx@beamer%
      \setbeamertemplate{navigation symbols}{}%
      \begin{frame}[plain]%
      \usebox\svg@box%
      \end{frame}%
    \else%
      \usebox\svg@box%
    \fi%
    \endgraf%
  \endgroup%
}
```

D. Processing Options

Setting the default options and processing the given ones during when loading the packages.

```
⟨*main⟩
\FamilyExecuteOptions{SVG}{%
  inkscape=true,inkscapeversion=auto,inkscapepath=basesubdir,%
  inkscapelatex=true,inkscapearea=drawing,distort=false,%
  usexcolor=true,usetransparent=true%
}
⟨/main⟩
⟨*extract⟩
\FamilyExecuteOptions{SVG}{%
  extract=true,extractpath=basesubdir,%
  extractruns=2,extractname=namenumbered,extractdistort=false,%
  convert=magick,convert=false,%
  gsdevice={png=png16m},gsdevice={jpeg=jpeg},gsdevice={jpg=jpeg},%
  gsdevice={tif=tiff48nc},gsdevice={tiff=tiff48nc},%
  gsdevice={eps=eps2write},gsdevice={ps=ps2write}%
}
⟨/extract⟩
\FamilyProcessOptions{SVG}
```

Index

Numbers written in italic refer to the page where the corresponding entry is described. Numbers underlined refer to the code line of the definition.

Change History

v1.0

General

- initial version by Philip Ilten 2

v2.00

General

- new maintainer: Falk Hanisch 2
- package **subfig** not required anymore 2
- re-implementation from scratch 2
- support of subfigures stopped due to the huge number of packages which deal with this topic and the large variety of implementing this functionality; naming exported graphics after their consecutive numbering can't be ensured for all variants of subfigures, so it's neglected 2

Implementation

- `clean` (opt.): changes, file list possible 1990
- `convert` (opt.): changed/extended 1510
- `convertdpi` (opt.): new 1613
- `convertformat` (opt.): new 1586
- `draft` (opt.): new 649
- `dvipsopt` (opt.): new 1461
- `end` (opt.): deprecated 1379
- `eps` (opt.): deprecated 1338
- `extract` (opt.): new 1306
- `extractapptex` (opt.): new 1954
- `extractformat` (opt.): new 1338
- `extractheight` (opt.): new 1858
- `extractname` (opt.): new 1818
- `extractpath` (opt.): new 1784
- `extractpreamble` (opt.): new 1379
- `extractpreambleend` (opt.): new 1379
- `extractpretex` (opt.): new 1954
- `extractruns` (opt.): new 1406
- `extractscale` (opt.): new 1858
- `extractwidth` (opt.): new 1858
- `gsdevice` (opt.): new 1751
- `gsexe` (opt.): new 1751
- `gsopt` (opt.): new 1751
- `height` (opt.): new 555
- `\includeinkscape`: new 761
- `\includesvg`:
  - changes, especially to optional parameters 709
  - `angle` (param.): new 744
  - `draft` (param.): new 712
  - `height` (param.): new 712
  - `inkscape` (param.): new 712
  - `inkscapearea` (param.): new 712
  - `inkscapedpi` (param.): new 712
  - `inkscapeformat` (param.): new 712
  - `inkscapelatex` (param.): new 712
  - `inkscapeopt` (param.): new 712
  - `lastpage` (param.): new 741
  - `origin` (param.): new 744
  - `scale` (param.): new 712
- `inkscape` (opt.): changed/extended 315
- `inkscapearea` (opt.): new 462
- `inkscapedpi` (opt.): new 476
- `inkscapeexe` (opt.): new 397
- `inkscapeformat` (opt.): new 438
- `inkscapelatex` (opt.): new 458
- `inkscapename` (opt.): new 549
- `inkscapeopt` (opt.): new 397
- `inkscapepath` (opt.): new 524
- `lastpage` (opt.): new 635
- `latexexe` (opt.): new 1423
- `latexext` (opt.): new 1423
- `latexopt` (opt.): new 1423
- `magickexe` (opt.): new 1717
- `magickoperator` (opt.): new 1717
- `magicksetting` (opt.): new 1717
- `name` (opt.):
  - deprecated 1818
  - support of **subfig** removed 1818
- `notransparent` (opt.): new 286
- `noxcolor` (opt.): new 286
- `off` (opt.): new 395, 1334
- `on` (opt.): new 395, 1334
- `path` (opt.): deprecated 1784
- `pdf` (opt.): deprecated 1338
- `pdflatex` (opt.): deprecated 1423
- `pdftoepsopt` (opt.): new 1461
- `pdftops` (opt.): deprecated 1461
- `pdftopsopt` (opt.): new 1461
- `png` (opt.): deprecated 1586
- `postex` (opt.): deprecated 612
- `preamble` (opt.): deprecated 1379
- `pstoepsopt` (opt.): new 1461
- `pstopdfopt` (opt.): new 1461
- `scale` (opt.): new 555
- `\setsvg`: deprecated 695
- `\svghidepreambleend`: new 2055
- `\svghidepreamblestart`: new 2055
- `\svgpath`: new 697
- `svgpath` (opt.): deprecated 505
- `\svgsetup`: new 695
- `usetransparent` (opt.): new 286
- `usexcolor` (opt.): new 286

v2.00a

Implementation

- `\svgxsetpapersize`: Bug fix for checking stock- and mediasizes 2536

v2.00b

Implementation

- `latex` (opt.): new, alternative key for `inkscapelatex` 458
- `tex` (opt.): new, alternative key for `inkscapelatex` 458

v2.01

General

- option `svgextension` added in order to change the format of files exported by ***Inkscape*** from `svg` to a custom one 6
- usage of `\input{`⟨*tex filename*⟩`}` within ***Inkscape*** graphics locates files in all declared searched folders 2

Implementation

- `\includesvg`:
  - `svgextension` (param.): new 712
- `inkscape` (opt.): using `\trim@spaces` 315

`\svg@append@input@path`: new . . . 1148
`\svg@get@path`:
using `\svg@set@input@path` 1189
using `\trim@spaces` 1189
`\svg@set@input@path`: new 1148
`svgextension` (opt.): new due to user request 517

v2.02

General
fixed errors with active double quotes from **babel** in path arguments *2*
multiple dots within file names possible *2*
option `distort` (or `keepaspectratio`) added for distortion of included graphics *6*
option `extractdistort` added for distortion of extracted graphics . . . *9*
package **trimspaces** required *2*
parameter `extractangle` for `\includesvg` and `\includeinkscape` implemented in order to rotate graphics during extractions *10*
Implementation
`distort` (opt.): new 555
`extractdistort` (opt.): new 1858
`extractname` (opt.): usage of `\svg@quotes@remove` 1818
`extractpath` (opt.): usage of `\svg@sanitize@dq` 1784
`\if@svgx@standalone`: new 2521
`\includeinkscape`:
usage of `\svg@extension@parse` . . 761
`extractangle` (param.): new 2033
`extractdistort` (param.): new . . 2033
`\includesvg`:
switched to `\svg@filename@parse` 709
`angle` (param.): validation of argument 744
`distort` (param.): new 712
`extractangle` (param.): new 2033
`extractdistort` (param.): new . . 2033
`inkscape` (opt.): usage of `\svg@sanitize@dq` 315
`inkscapepath` (opt.): usage of `\svg@sanitize@dq` 524
`keepaspectratio` (opt.): new 555
`\svg@append@input@path`: avoid duplicates in `\input@path` 1148
`\svg@deactivate@dq`: new 46
`\svg@extension@parse`: new 143
`\svg@extension@@parse`: new 143
`\svg@file@missing`: notify svg file when missing exported files 1014
`\svg@filename@parse`:
usage of `\svg@extension@parse` . . 106
usage of `\svg@remove@leadingchar` 106
usage of `\svg@sanitize@dq` 106
`\svg@local@param@set`: reasonable value for key `distort` 655
`\svg@normalize@path`: usage of `\svg@deactivate@dq` 238
`\svg@quotes@remove`:
calling `\svg@quotes@check` 65
usage of `\svg@sanitize@dq` 65
`\svg@remove@leadingchar`: new 90
`\svg@sanitize@dq`: new 51
`\svg@set@input@path`: usage of `\svg@deactivate@dq` 1148
`svgextension` (opt.):
usage of `\svg@quotes@remove` 517
usage of `\svg@remove@leadingchar` 517
`\svgpath`: parse argument for enclosing braces and provide those if necessary 697
`\svgx@setbox`: new 2521
`\svgxsetbox`: late execution of `\svgxsetpapersize` 2521

v2.02a

General
fix bug for package **polyglossia** which fakes **babel** poorly *2*
Implementation
`\svg@deactivate@dq`: bug fix for **polyglossia** 46

v2.02b

General
fix bug for package **tikzscale** which changes `\includegraphics` globally *2*
Implementation
`\svg@patches`: fix bug for package **tikzscale**: store original definitions of `\picture` and `\includegraphics` right after loading package **svg** . 1229

v2.02c

General
fix bugs with kernel (2019/10/01) regarding file name parsing *2*

v2.02d

General
conditional invocation of ***Inkscape*** export based on file modification date implemented for XeTEX *2*
fix bugs with kernel (2019/10/01) regarding file name parsing, see `https://github.com/mrpiggi/svg/issues/16` *2*
Implementation
`\svg@iffilenewer`: use `\filemoddate` with XeLaTeX, see `https://github.com/mrpiggi/svg/issues/12` 179

v2.02f

General
command line interface of ***Inkscape 1.0*** is supported *2*
option inkscapeversion for explicitly setting ***Inkscape***-CLI version; automatic detection implemented . . *5*
Implementation
`\if@svg@ink@ver@detect`: new . . . 1060
`inkscapeexe` (opt.): only usable in preamble 397
`inkscapeversion` (opt.): new 397
`\svg@ink@cmd`: distinguish ***Inkscape*** CLI versions 493
`\svg@ink@ver`: new 397
`\svg@ink@ver@detect`: new 1060

`\svg@ink@ver@settings`: new 1060
`\svg@@ink@ver@detect`: new 1060

v2.02g

General
fix for multiple dots in file names (#27) *2*
Implementation
`\svg@@input`: parsing of file extension discarded; meanwhile taken over by the kernel 901

v2.02h

General
fix for package **transparent** (#28) *2*

v2.02i

General
update for package **scrlfile** v3.32 and kernel (2020/10/01) *2*

v2.02j

General
bug fix for automatic version detection of ***Inkscape*** with MiKTEX *2*
Implementation
`\svg@ink@ver@explore`: new 1126

v2.02k

General
MiKTEXwas updated, bug fix removed . *2*