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

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

v2.02b (2018/11/12)

The **svg** package is intended for the automated integration of SVG graphics into LaTeX documents. Therefor the capabilities provided by ***Inkscape***—or more precisely its command line tool—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 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.

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.

Both packages **svg** and **svg-extract** rely heavily upon executing commands from the shell using the `\ShellEscape` command—or respectively the old known `\write18`—for executing a variety of commands directly to the system. So it is necessary to include the flag `--shell-escape` when compiling documents using **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*.

---

[1] http://www.ctan.org/pkg/svg-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 functionallity. These are:

scrbase for the definition and handling of options in key-value-syntax
ifpdf, ifluatex, ifxetex for flow control depending on the used LATEX engine
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 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 similiar 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 assinged to `usexcolor` and `usetransparent`, while `noxcolor` and `notransparent` don't accept any value.

```
\usepackage[⟨options⟩]{svg}
```

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. Settings with `\svgsetup` are done in the current scope which means globally or within the current group.

```
\svgsetup{⟨options⟩}
```

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

\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 terminate with / last. 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's recommended to avoid any spaces and/or quotes respectively `\dq` both in paths and file names, especially when DVI output is active.

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. Unfortunately a primitive like `\pdffilemoddate` is missing for XeTeX, so with this engine, the behaviour will be the same as `inkscape=forced`.

`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 subfolder `./svg-inkscape/` within the current working directory is used by default (`inkscapepath=basesubdir`).

`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.

inkscapename (opt.)   The file names of the **Inkscape** export are derived from the name of the base SVG file and can be modified with inkscapename=⟨*filename*⟩. It's possible to use counters for specifying the name of the exported file. Repeatedly specifying the same file name will overwrite previously created files.

inkscapeexe (opt.)   For including a SVG file, **Inkscape** is used to separate the text and image from the SVG file itself. In order to execute the command line tool from shell, the path where the executable is located has to be known to the operating system. You can check this by typing `inkscape -V` into the shell. If this check fails and you don't want to change environment variable `path` on your OS, you can use option inkscapeexe to set the absolute path where the executable of **Inkscape** is located. The option is set to inkscapeexe=inkscape by default.

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 seperate 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.

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 tool. For further information see the documentation of **Inkscape**[2].

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.[3] 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⟩}
```

---

[2]https://inkscape.org/de/doc/inkscape-man.html
[3]to provide compatibility for package **graphicx**, it's possible to use keepaspectratio=true as alias for distort=false and the other way round

draft (opt.) · This option can be used with booelan 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[4] 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 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.) · 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 any spaces and/or quotes respectively `\dq` both in paths an file names. Espacially when DVI output is active using quotes will certainly cause an error.

width (param.)
height (param.)
distort (param.)
scale (param.)
pretex (param.)
apptex (param.)
draft (param.) · 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.[5] 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*⟩`}`

---

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

[5] Due to the lack of XeTeX to compare file modification dates, using this LaTeX engine leads to *Inkscape* exports with every run unless `inkscape=false` is used.

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 2ε distributions. In additon, the command line tools 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 of 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 implemetations, this feature can't be provided reliably by **svg-extract**. See option extractname for further information.

3.1. General settings

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.

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

With this 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]{⟨filename⟩}
```

7

All option described below can be used toghether with \svgsetup 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 parameters have an effect only once when the specific command is executed and remain unchanged afterwards. These parameters are: `extract`, `extractpreamble`, `extractformat`, `extractruns`, `latexopt`, `extractwidth`, `extractheight`, `extractdistort`, `extractscale`, `extractangle`, `extractpretex`, `extractapptex`, `convert`, `convertformat`, `convertdpi`, `magicksetting`, `magickoperator`, `gsopt`, `gsdevice`, `clean`, `exclude`.

3.2. Extract independent grahic 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 `extractpath=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`.

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 count of extracted files is used instead of the suffix.

numbered/section/numberedsection/sectionnumbered
> The file name is composed by the number of extracted files and the current outline numbering.

⟨filename⟩
> You can use any file name, the file extension is derived from option `extractformat`. 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 a 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. Only option `extractdistort` is initialized to `false` and does not inherit from `distort` by default.

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 grahic files by compiling the generated auxiliary LATEX file, it's maybe necessary to do multiple LATEX runs on this file. The number of runs can be controlled with option `extractruns`. It's set to `extractruns=2` by default.
latexexe (opt.) latexopt (opt.) latexext (opt.)	For the extraction of an independent grahic file, the LATEX program is used which is set by the `latexexe` option. Depending on the LATEX processor 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 runs, which are performed during the extraction process by the `latexopt` option. If you are used to utilize a other file 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 processor, 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 2ε 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).

3.3. Convert extracted grahic files

Based on the extraction of independent graphic files, the **svg-extract** packages also provides the possibility to convert those 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.

`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 $\langle option \rangle = \langle value \rangle$ or $\langle option \rangle = \{\langle outputformat \rangle = \langle value \rangle\}$ and even $\langle option \rangle = \{\langle outputformat \rangle += \langle value \rangle\}$ 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 a 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 ($\langle option \rangle = $`\relax`) or a specific one ($\langle option \rangle = \{\langle outputformat \rangle$`[+]`$=$`\relax`$\}$).

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 tool 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 tool[6].

3.3.2. Settings for the invocation of *Ghostscript*

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

The conversion with **Ghostscript** is done with command-line tool `gs` on unix-like operating systems and `gswin64c` or `gswin32c` on Windows. 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].

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}
```

---

[6] http://www.imagemagick.org/script/command-line-processing.php
[7] https://ghostscript.com/doc/current/Use.htm
[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.
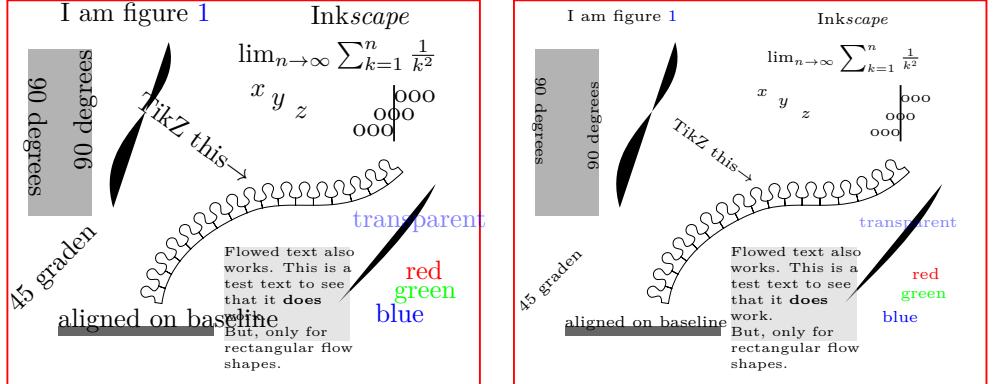
(a) This text is too large! (b) This text fits better.

Figure 1: An example figure with LaTeX support
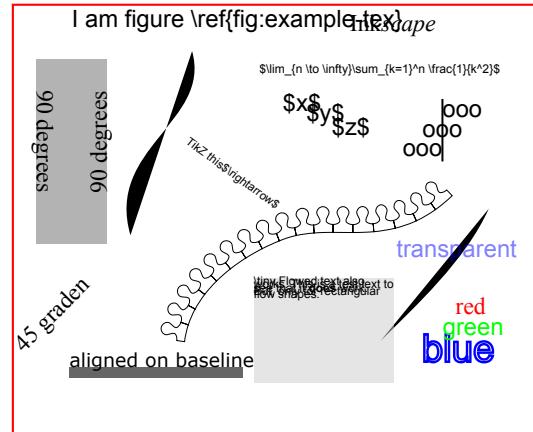

Figure 2: The same example figure without LaTeX support

```
\usepackage{subcaption}
\begin{document}
\begin{figure}
  \begin{minipage}{.5\linewidth}
    \includesvg[width=\linewidth]{svg-example}%
    \subcaption{This text is too large!}
  \end{minipage}%
  \begin{minipage}{.5\linewidth}
    \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}
```

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/examples/. Second, you have to run the desired LaTeX engine with `--shell-escape` option enabled.

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.

As you can see, Figure 1a is created with default settings, except for the width specification.

So the **Inkscape** export with LaTeX support is done as well as the extraction of a independent graphic file in PDF format as the **svg-extract** package was loaded.

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 via 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 from shell 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

---

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

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/examples/, 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);

  // 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.
```

---

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

$$z(x, y) = \frac{1}{\sigma_x \sigma_y \sqrt{4\pi^2}} \exp\left(-\left(\frac{(x-\mu_x)^2}{2\sigma_x^2} + \frac{(y-\mu_y)^2}{2\sigma_y^2}\right)\right)$$
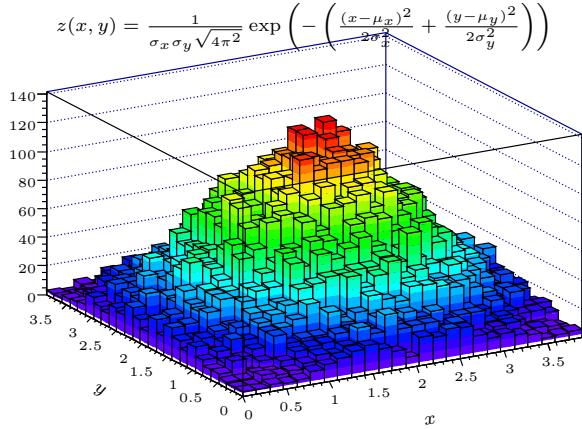
Figure 3: Rendering of a **ROOT** plot—no more *Comic CERNs*

```
  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!

15

Part II.
Implementation

A. Initialization

A.1. Packages

The package **svg** requires **scrbase** for options processing, the packages **ifluatex**, **ifpdf** and **ifxetex** for detecting the used LATEX engine, **pdftexcmds** for pdfTEX primitives when using LuaTEX, **shellesc** and **ifplatform** for engine independent access to systems commands and files as well as **graphicx** for the inclusion of PDF files. The usage of packages **xcolor** and **transparent** can be switched of with the corresponding options. Package **svg-extract** only needs package **svg** itself.

```
1 ⟨*base⟩
2 \RequirePackage{scrbase}[2016/06/14]
3 \RequirePackage{ifpdf}[2016/05/14]
4 \RequirePackage{ifluatex}[2016/05/16]
5 \RequirePackage{ifxetex}[2010/09/12]
6 \RequirePackage{pdftexcmds}[2016/05/21]
7 \RequirePackage{shellesc}[2016/06/07]
8 \RequirePackage{trimspaces}[2009/09/17]
9 \RequirePackage{graphicx}[1999/02/16]
10 ⟨/base⟩
11 ⟨*extract⟩
12 \RequirePackage{svg}[2017/03/27]
13 ⟨/extract⟩
```

A.2. Helper macros

\svg@tempa
\svg@tempb
\svg@box
\if@svg@tempswa

Internal temporary macros. The catcode for double quotes are also temporarily changed.

```
14 \newcommand*\svg@tempa{}
15 \newcommand*\svg@tempb{}
16 \newbox\svg@box
17 \newif\if@svg@tempswa
18 \edef\svg@catcodecodes@restore{%
19   \catcode`\noexpand\"\the\catcode`\"\relax%
20 }
21 \@makeother\"%
```

B. Including SVG files with package svg

B.1. Options

All options, which can be set either as package options or with \svgsetup, as well as the optional parameters for both user commands \includesvg[⟨*parameters*⟩]{⟨*svg filename*⟩} and \includeinkscape[⟨*parameters*⟩]{⟨*graphic filename*⟩} are defined with the interface provided by package **scrbase**.

```
22 \DefineFamily{SVG}
23 \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.

```
24 \newcommand*\svg@deprecated@key[3][svg]{%
25   \PackageWarning{#1}{%
26     The option key '#2' is deprecated.\MessageBreak%
27     It's recommended to use '#3'\MessageBreak%
28     instead%
29   }%
30   \FamilyOptions{SVG}{#3}%
31 }
```

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 anyways.

usexcolor (opt.)   Options for preventing packages **xcolor** and **transparent** to be loaded.
noxcolor (opt.)
\if@svg@use@xcolor
usetransparent (opt.)
notransparent (opt.)
\if@svg@use@transparent

```
32 \newif\if@svg@use@xcolor
33 \FamilyBoolKey{SVG}{usexcolor}{@svg@use@xcolor}
34 \DeclareOption{noxcolor}{\FamilyOptions{SVG}{usexcolor=false}}
35 \newif\if@svg@use@transparent
36 \FamilyBoolKey{SVG}{usetransparent}{@svg@use@transparent}
37 \DeclareOption{notransparent}{\FamilyOptions{SVG}{usetransparent=false}}
```

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

```
38 \AtEndOfPackage{%
39   \RelaxFamilyKey{SVG}{usexcolor}%
40   \RelaxFamilyKey{SVG}{usetransparent}%
41   \if@svg@use@xcolor%
42     \RequirePackage{xcolor}[2016/05/11]%
43   \else%
44     \AfterPackage*{xcolor}{%
45       \PackageWarning{svg}{Package 'xcolor' was loaded anyway}%
46     }%
47   \fi%
48   \if@svg@use@transparent%
49     \RequirePackage{transparent}[2016/05/16]%
50   \else%
51     \AfterPackage*{transparent}{%
52       \PackageWarning{svg}{Package 'transparent' was loaded anyway}%
53     }%
54   \fi%
55 }
```

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.)   The intension of option inkscape is to control the running behaviour of **Inkscape**. It can
\svg@ink@mode   be switched off at all (inkscape=false) or invoked only if necessary (inkscape=true) or the command line call can 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.

```
56 \newcommand*\svg@ink@mode{}
57 \DefineFamilyKey{SVG}{inkscape}[true]{%
58   \lowercase{\svg@sanitize@dq\svg@tempb{#1}}%
59   \FamilySetNumerical{SVG}{inkscape}{svg@tempa}{%
60     {false}{0},{off}{0},{no}{0},%
```

17

```
61      {true}{1},{on}{1},{yes}{1},{onlynewer}{1},{newer}{1},%
62      {force}{2},{forced}{2},{overwrite}{2},%
63      {pdf}{3},{eps}{4},{ps}{5},{png}{6},%
64      {drawing}{7},{crop}{7},%
65      {page}{8},{nocrop}{8},%
66      {tex}{9},{latex}{9},{exportlatex}{9},{latexexport}{9},%
67      {notex}{10},{nolatex}{10},{noexportlatex}{10},{nolatexexport}{10},%
68      {latexnoexport}{10},{raw}{10},{plain}{10},{simple}{10}%
69    }{\svg@tempb}%
70    \ifx\FamilyKeyState\FamilyKeyStateProcessed%
```

Setting the mode for invoking **Inkscape**...

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

...and the part as wrapper for different options.

```
74        \ifcase\svg@tempa\relax\or\or\or% pdf
75          \FamilyOptions{SVG}{inkscapeformat=pdf}%
76        \or% eps
77          \FamilyOptions{SVG}{inkscapeformat=eps}%
78        \or% ps
79          \FamilyOptions{SVG}{inkscapeformat=ps}%
80        \or% png
81          \FamilyOptions{SVG}{inkscapeformat=png}%
82        \or% drawing
83          \FamilyOptions{SVG}{inkscapearea=drawing}%
84        \or% page
85          \FamilyOptions{SVG}{inkscapearea=page}%
86        \or% tex
87          \FamilyOptions{SVG}{inkscapelatex=true}%
88        \or% notex
89          \FamilyOptions{SVG}{inkscapelatex=false}%
90        \fi%
91      \fi%
```

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

```
92    \else% dpi
93      \def\svg@tempa##1dpi##2\@nil{%
94        \ifstr{##2}{dpi}{\FamilyOptions{SVG}{inkscapedpi=##1}}{}%
95      }%
96      \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.

```
97        \ifx\FamilyKeyState\FamilyKeyStateProcessed\else%
```

Splitting executable from options with delimitted 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`.

```
98        \svg@quotes@remove[{#1}]{\svg@tempb}%
99        \def\svg@tempa##1-##2\@nil{%
100         \IfArgIsEmpty{##2}{\def\svg@tempb{}}{%
101           \def\svg@tempa####1-\@nil{\def\svg@tempb{-####1}}%
102           \svg@tempa##2\@nil%
103         }%
104         \edef\svg@tempa{\trim@spaces{##1}}%
105       }%
106       \edef\svg@tempb{%
107         \noexpand\svg@tempa\svg@tempb-\noexpand\@nil%
108       }%
```

```
109        \svg@tempb%
110        \if@svg@quotes@found%
111          \edef\svg@tempa{"\svg@tempa"}%
112        \fi%
113        \PackageWarning{svg}{%
114          Setting the executable%
115          \ifx\svg@tempb\@empty\else%
116            \space and associated options%
117          \fi%
118          \MessageBreak%
119          for Inkscape should be done with options\MessageBreak%
120          `inkscapeexe=\svg@tempa'%
121          \ifx\svg@tempb\@empty\else%
122            \MessageBreak and `inkscapeopt=\svg@tempb'%
123          \fi.\MessageBreak%
124          Nevertheless, this was done by now anyway%
125        }%
126        \edef\svg@tempa{%
127          \noexpand\FamilyOptions{SVG}{inkscapeexe=\svg@tempa}%
128          \ifx\svg@tempb\@empty\else%
129            \noexpand\FamilyOptions{SVG}{inkscapeopt=\svg@tempb}%
130          \fi%
131        }%
132        \svg@tempa%
133      \fi%
134    \fi%
135 }
```

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

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

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.

```
138 \newcommand*\svg@ink@format{pdf}
139 \ifxetex\else\ifpdf\else
140   \renewcommand*\svg@ink@format{eps}
141 \fi\fi
142 \DefineFamilyKey{SVG}{inkscapeformat}{%
143   \lowercase{\def\svg@tempa{#1}}%
144   \FamilySetNumerical{SVG}{inkscapeformat}{svg@tempa}{%
145     {pdf}{0},{eps}{1},{ps}{2},{png}{3}%
146   }{\svg@tempa}%
147   \ifx\FamilyKeyState\FamilyKeyStateProcessed%
148     \ifcase\svg@tempa\relax% latex
149       \renewcommand*\svg@ink@format{pdf}%
150     \or% eps
151       \renewcommand*\svg@ink@format{eps}%
152     \or% ps
153       \renewcommand*\svg@ink@format{ps}%
154     \or% png
155       \renewcommand*\svg@ink@format{png}%
156     \fi%
157   \fi%
158 }
```

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 seperate LaTeX file.

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

```
161 \FamilyBoolKey{SVG}{latex}{@svg@ink@latex}
162 \FamilyBoolKey{SVG}{tex}{@svg@ink@latex}
```

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

```
163 \newcommand*\svg@ink@area{}
164 \DefineFamilyKey{SVG}{inkscapearea}{%
165   \FamilySetNumerical{SVG}{inkscapearea}{svg@tempa}{%
166     {drawing}{0},{crop}{0},%
167     {page}{1},{nocrop}{1}%
168   }{#1}%
169   \ifx\FamilyKeyState\FamilyKeyStateProcessed%
170     \ifcase\svg@tempa\relax% drawing
171       \renewcommand*\svg@ink@area{-D}%
172     \else% page
173       \renewcommand*\svg@ink@area{-C}%
174     \fi%
175   \fi%
176 }
```

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

```
177 \newcommand*\svg@ink@dpi{}
178 \let\svg@ink@dpi\relax
179 \DefineFamilyKey{SVG}{inkscapedpi}{%
180   \FamilyKeyStateUnknownValue%
181   \svg@ifvalueisrelax{#1}{%
182     \let\svg@ink@dpi\relax%
183     \FamilyKeyStateProcessed%
184   }{%
185     \def\svg@tempa##1dpi##2\@nil{\def\svg@tempa{##1}}%
186     \lowercase{\svg@tempa#1dpi\@nil}%
187     \ifnumber{\svg@tempa}{%
188       \edef\svg@ink@dpi{\svg@tempa}%
189       \FamilyKeyStateProcessed%
190     }{}%
191   }%
192 }
193 \DefineFamilyKey{SVG}{inkscapedensity}{\FamilyOptions{SVG}{inkscapedpi=#1}}
```

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

```
194 \newcommand*\svg@ink@exe{inkscape}
195 \DefineFamilyKey{SVG}{inkscapeexe}{%
196   \renewcommand*\svg@ink@exe{#1}%
197   \FamilyKeyStateProcessed%
198 }
199 \newcommand*\svg@ink@opt{}
200 \DefineFamilyKey{SVG}{inkscapeopt}{%
201   \renewcommand*\svg@ink@opt{#1}%
202   \FamilyKeyStateProcessed%
203 }
```

B.1.2. Setting input folder and file

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

```
204 \DefineFamilyKey{SVG}{svgpath}{%
205   \PackageWarning{svg}{%
206     The key 'svgpath' is deprecated. It's recommended\MessageBreak%
207     to use '\string\svgpath' instead%
```

```
208    }%
209    \ifx\svgpath\@undefined%
210      \AtEndOfPackage{\svgpath{{#1}}}%
211    \else%
212      \svgpath{{#1}}%
213    \fi%
214    \FamilyKeyStateProcessed%
215 }
```

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

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

The extension should be in lower case letters.

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

Remove leading dots from the extension.

```
219    \svg@remove@leadingchar.\svg@file@ext%
220 }
221 \DefineFamilyKey{SVG}{extension}{\FamilyOptions{SVG}{svgextension=#1}}
222 \DefineFamilyKey{SVG}{ext}{\FamilyOptions{SVG}{svgextension=#1}}
```

B.1.3. Setting output folder

The option inkscapepath controls, in which folder the results of the **Inkscape** export will be located. With option inkscapename the name of the exported file itself can be changed.

```
223 \newcommand*\svg@out@path{}
224 \newcommand*\svg@out@name{\svg@file@name\svg@file@suffix}
225 \newcommand*\svg@out@base{\svg@out@path\svg@out@name.\svg@ink@format}
226 \DefineFamilyKey{SVG}{inkscapepath}{%
227    \svg@sanitize@dq\svg@tempb{#1}%
228    \FamilySetNumerical{SVG}{inkscapepath}{svg@tempa}{%
229      {svgpath}{0},{svgdir}{0},%
230      {svgsubpath}{1},{svgsubdir}{1},%
231      {basepath}{2},{basedir}{2},{jobpath}{2},{jobdir}{2},%
232      {basesubpath}{3},{basesubdir}{3},{jobsubpath}{3},{jobsubdir}{3}%
233    }{\svg@tempb}%
234    \ifx\FamilyKeyState\FamilyKeyStateProcessed%
235      \ifcase\svg@tempa\relax% svgpath
236        \renewcommand*\svg@out@path{\svg@file@path}%
237      \or% svgsubpath
238        \renewcommand*\svg@out@path{\svg@file@path svg-inkscape/}%
239      \or% basepath
240        \renewcommand*\svg@out@path{./}%
241      \or% basesubpath
242        \renewcommand*\svg@out@path{./svg-inkscape/}%
243      \fi%
244    \else%
245      \edef\svg@out@path{\svg@tempb}%
246      \svg@normalize@path{\svg@out@path}%
247      \FamilyKeyStateProcessed%
248    \fi%
249 }
250 \DefineFamilyKey{SVG}{inkscapename}{%
251    \renewcommand*\svg@out@name{#1\svg@file@suffix}%
252    \FamilyKeyStateProcessed%
253 }
```

21

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.

```
254 \newcommand*\svg@param@width{\z@}
255 \DefineFamilyKey{SVG}{width}{%
256   \FamilyKeyStateUnknownValue%
257   \svg@ifvalueisrelax{#1}{%
258     \renewcommand*\svg@param@width{\z@}%
259     \FamilyKeyStateProcessed%
260   }{%
261     \FamilySetLengthMacro{SVG}{width}{\svg@param@width}{#1}%
262     \ifx\FamilyKeyState\FamilyKeyStateProcessed%
263       \ifdim\svg@param@width<\z@\relax%
264         \FamilyKeyStateUnknownValue%
265       \fi%
266     \fi%
267   }%
268 }
269 \newcommand*\svg@param@height{\z@}
270 \DefineFamilyKey{SVG}{height}{%
271   \FamilyKeyStateUnknownValue%
272   \svg@ifvalueisrelax{#1}{%
273     \renewcommand*\svg@param@height{\z@}%
274     \FamilyKeyStateProcessed%
275   }{%
276     \FamilySetLengthMacro{SVG}{height}{\svg@param@height}{#1}%
277     \ifx\FamilyKeyState\FamilyKeyStateProcessed%
278       \ifdim\svg@param@height<\z@\relax%
279         \FamilyKeyStateUnknownValue%
280       \fi%
281     \fi%
282   }%
283 }
284 \newif\if@svg@param@distort
285 \FamilyBoolKey{SVG}{distort}{@svg@param@distort}
286 \DefineFamilyKey{SVG}{keepaspectratio}[true]{%
287   \FamilySetBool{SVG}{keepaspectratio}{@svg@tempswa}{#1}%
288   \ifx\FamilyKeyState\FamilyKeyStateProcessed%
289     \if@svg@tempswa%
290       \FamilyOptions{SVG}{distort=false}%
291     \else
292       \FamilyOptions{SVG}{distort=true}%
293     \fi%
294   \fi%
295 }
296 \newcommand*\svg@param@scale{1}
297 \DefineFamilyKey{SVG}{scale}{%
298   \FamilyKeyStateUnknownValue%
299   \svg@ifvalueisrelax{#1}{%
300     \renewcommand*\svg@param@scale{1}%
301     \FamilyKeyStateProcessed%
302   }{%
303     \ifisdimension{#1\p@}{%
304       \ifdim\dimexpr#1\p@\relax>\z@\relax%
305         \renewcommand*\svg@param@scale{#1}%
306         \FamilyKeyStateProcessed%
307       \fi%
308     }{}%
309   }%
310 }
```

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.

```
311 \newcommand*\svg@param@pretex{}
312 \let\svg@param@pretex\relax
313 \DefineFamilyKey{SVG}{pretex}{%
314   \svg@ifvalueisrelax{#1}{%
315     \let\svg@param@pretex\relax%
316   }{%
317     \def\svg@param@pretex{#1}%
318   }%
319   \FamilyKeyStateProcessed%
320 }
321 \newcommand*\svg@param@apptex{}
322 \let\svg@param@apptex\relax
323 \DefineFamilyKey{SVG}{apptex}{%
324   \svg@ifvalueisrelax{#1}{%
325     \let\svg@param@apptex\relax%
326   }{%
327     \def\svg@param@apptex{#1}%
328   }%
329   \FamilyKeyStateProcessed%
330 }
331 \DefineFamilyKey{SVG}{postex}{%
332   \svg@deprecated@key{postex=#1}{apptex=#1}%
333 }
```

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.

```
334 \newcounter{svg@param@lastpage}
335 \DefineFamilyKey{SVG}{lastpage}{%
336   \FamilySetNumerical{SVG}{lastpage}{svg@tempa}{%
337     {false}{0},{off}{0},{no}{0},{ignore}{0},%
338     {true}{1},{on}{1},{yes}{1},{auto}{1}%
339   }{#1}%
340   \ifx\FamilyKeyState\FamilyKeyStateProcessed%
341     \ifcase\svg@tempa\relax% false
342       \FamilySetCounter{SVG}{lastpage}{svg@param@lastpage}{\m@ne}%
343     \or% true
344       \FamilySetCounter{SVG}{lastpage}{svg@param@lastpage}{\z@}%
345     \fi%
346   \fi%
347 }
```

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

```
348 \newif\if@svg@draft
349 \FamilyBoolKey{SVG}{draft}{@svg@draft}
350 \AtBeginDocument{\if@svg@draft\else\ifGin@draft\@svg@drafttrue\fi\fi}
```

B.2. Handling path information

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.

In order to avoid errors concerning file names with package **babel** and it's active double quotes, this command is defined.

```
351 \newcommand*\svg@deactivate@dq{}
```

```
352 \AfterPackage+{babel}{%
353   \renewcommand*\svg@deactivate@dq{\bbl@deactivate{"}}%
354   \providecommand*\bbl@deactivate[1]{}%
355 }
```

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

```
356 \newcommand*\svg@sanitize@dq[2]{%
357   \begingroup%
358     \svg@deactivate@dq%
359     \edef\svg@tempa{\endgroup\def\noexpand#1{#2}}%
360   \svg@tempa%
361 }
```

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

```
362 \newcommand*\svg@quotes@remove[2][]{%
363   \begingroup%
364     \IfArgIsEmpty{#1}{\def\svg@tempb{#2}}{\def\svg@tempb{#1}}%
365     \svg@sanitize@dq\svg@tempa{\svg@tempb}%
366     \expandafter\svg@quotes@check\expandafter{\svg@tempa}%
367     \expandafter\svg@quotes@@remove\svg@tempa""\@nil%
368     \edef\svg@tempb{%
369       \endgroup%
370       \def\noexpand#2{\svg@tempa}%
371       \if@svg@quotes@found%
372         \noexpand\@svg@quotes@foundtrue%
373       \else%
374         \noexpand\@svg@quotes@foundfalse%
375       \fi%
376     }%
377   \svg@tempb%
378 }
379 \newcommand*\svg@quotes@@remove{}
380 \def\svg@quotes@@remove#1"#2"#3\@nil{%
381   \IfArgIsEmpty{#2}{%
382     \edef\svg@tempa{#1}%
383   }{%
384     \svg@quotes@@remove#1#2#3""\@nil%
385   }%
386 }
```

\svg@quotes@check   During the treatment of paths, it may be necessary to temporarily remove quotes and, if
\svg@quotes@@check   required, add them again later. For this purpose, the switch \if@svg@quotes@found as
\if@svg@quotes@found   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.

```
387 \newif\if@svg@quotes@found
388 \newcommand*\svg@quotes@check[1]{%
389   \expandafter\svg@quotes@@check#1"\@nil%
390 }
391 \newcommand*\svg@quotes@@check{}
392 \def\svg@quotes@@check#1"#2\@nil{%
393   \IfArgIsEmpty{#2}{\@svg@quotes@foundfalse}{\@svg@quotes@foundtrue}%
394 }
```

\svg@remove@leadingchar   This command removes the single character in given with the first argument from the
                           expanded macro in the second argument.

```
395 \newcommand*\svg@remove@leadingchar[2]{%
396   \begingroup%
```

```
397    \svg@sanitize@dq\svg@tempa{#2}%
398    \def\svg@tempb{%
399      \def\svg@tempa####1\@nil{\def\svg@tempa{####1}}%
400      \kernel@ifnextchar#1%
401        {\expandafter\svg@tempa\@gobble}%
402        {\svg@tempa}%
403    }%
404    \expandafter\svg@tempb\svg@tempa\@nil%
405    \edef\svg@tempb{%
406      \endgroup%
407      \def\noexpand#2{\svg@tempa}%
408    }%
409  \svg@tempb%
410 }
```

\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.

```
411 \newcommand*\svg@set@input@path[1][]{%
412   \begingroup%
413     \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.

```
414     \ifx\svg@file@path\@empty\else%
415       \svg@normalize@path{\svg@file@path}%
416       \edef\svg@file@path{{\svg@file@path}}%
417     \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.

```
418     \svg@append@input@path{\svg@file@path}{\svg@input@path}%
419     \svg@append@input@path{\svg@file@path}{\Ginput@path}%
420     \IfArgIsEmpty{#1}{}{\svg@append@input@path{\svg@file@path}{{#1}}}%
421     \svg@append@input@path{\svg@file@path}{\input@path}%
```

Finally, \input@path is set.

```
422     \edef\svg@tempa{%
423       \endgroup%
424       \ifx\svg@file@path\@empty\else%
425         \def\noexpand\input@path{\svg@file@path}%
426       \fi%
427     }%
428   \svg@tempa%
429 }
```

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.

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

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

```
435        \ifx\svg@tempa\@empty\else%
436          \@svg@tempswatrue%
437          \svg@normalize@path{\svg@tempa}%
438          \expandafter\@tfor\expandafter\svg@tempb\expandafter:\expandafter=%
439            #1\do{%
440            \ifx\svg@tempa\svg@tempb%
441              \@svg@tempswafalse%
442              \@break@tfor%
443            \fi%
444          }%
445          \if@svg@tempswa%
446            \edef#1{#1{\svg@tempa}}%
447          \fi%
448        \fi%
449      }%
450    \fi%
451 }
```

`\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`.

```
452 \newcommand*\svg@normalize@path[1]{%
453    \begingroup%
454      \svg@quotes@remove[{#1}]{\svg@tempa}%
455      \ifx\svg@tempa\@empty\relax%
456        \def\svg@tempa{./}%
457      \fi%
458      \expandafter\svg@normalize@@path\svg@tempa//\@nil%
459      \edef\svg@tempb{%
460        \endgroup%
461        \if@svg@quotes@found%
462          \def\noexpand#1{"\svg@tempa"}%
463        \else%
464          \def\noexpand#1{\svg@tempa}%
465        \fi%
466      }%
467    \svg@tempb%
468 }
469 \newcommand*\svg@normalize@@path{}
470 \def\svg@normalize@@path#1/#2/\@nil{%
471    \IfArgIsEmpty{#2}{%
472      \IfArgIsEmpty{#1}{\def\svg@tempa{}}{\def\svg@tempa{#1/}}%
473    }{%
474      \svg@normalize@@path#2/\@nil%
475      \edef\svg@tempa{#1/\svg@tempa}%
476    }%
477 }
```

`\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 reseting the key. Therefore, `\svg@ifvalueisrelax` checks, whether `\relax` was used as value or not.

```
478 \newcommand*\svg@ifvalueisrelax[1]{%
479    \begingroup%
480      \def\svg@tempa{#1}%
481      \def\svg@tempb{\relax}%
482      \ifx\svg@tempa\svg@tempb\relax%
483        \aftergroup\@firstoftwo%
484      \else%
485        \aftergroup\@secondoftwo%
486      \fi%
```

```
487      \endgroup%
488 }
```

\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.

```
489 \newif\if@svg@file@found
490 \newcommand*\svg@file@path{}
491 \newcommand*\svg@file@name{}
492 \newcommand*\svg@file@base{}
493 \newcommand*\svg@file@suffix{}
494 \newcommand*\svg@get@path[3][\svg@file@ext]{%
495    \begingroup%
```

A maybe given, unneeded file extension is removed.

```
496      \svg@filename@parse[{#1}]{#2}%
497      \IfArgIsEmpty{#1}{%
498        \edef\svg@tempa{\filename@area\filename@base.\filename@ext}%
499      }{%
500        \edef\svg@tempa{\filename@area\filename@base.#1}%
501      }%
```

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.

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

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

```
503      \@svg@tempswafalse%
504      \expandafter\IfFileExists\expandafter{\svg@tempa}{%
505        \@svg@tempswatrue%
506        \edef\@filef@und{\expandafter\trim@spaces\expandafter{\@filef@und}}%
507        \svg@filename@parse[{#1}]{\@filef@und}%
508      }{}%
509      \edef\svg@tempa{%
510        \endgroup%
511        \if@svg@tempswa%
512          \noexpand\@svg@file@foundtrue%
513          \def\noexpand\svg@file@path{\filename@area}%
514          \def\noexpand\svg@file@name{\filename@base}%
515          \def\noexpand\svg@file@base{\filename@area\filename@base}%
516        \else%
517          \noexpand\@svg@file@foundfalse%
518          \def\noexpand\svg@file@path{}%
519          \def\noexpand\svg@file@name{#2}%
520          \def\noexpand\svg@file@base{#2}%
521        \fi%
522      }%
523    \svg@tempa%
524 }
```

\svg@filename@parse

As the internal LaTeX 2ε 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.

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

27

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

```
527     \svg@sanitize@dq\svg@tempa{#2}%
528     \expandafter\filename@parse\expandafter{\svg@tempa}%
529 % If there are quotes in the file path, the closing one will be found as first
530 % character in \cs{filename@base} as \cs{filename@area} is splitted at the last
531 % slash. This leading quote is removed from \cs{filename@base} with
532 % \cs{svg@remove@leadingchar}.
533 %     \begin{macrocode}
534     \svg@quotes@remove{\filename@area}%
535     \if@svg@quotes@found%
536       \edef\filename@area{"\filename@area"}%
537       \svg@remove@leadingchar"\filename@base%
538     \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`.

```
539     \ifx\filename@ext\relax\else%
540       \svg@quotes@remove{\filename@ext}%
541       \svg@extension@parse{#1}%
542       \if@svg@quotes@found%
543         \edef\filename@base{\filename@base"}%
544       \fi%
545     \fi%
```

Quotes within `\filename@base` are normalized.

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

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

```
550     \edef\svg@tempa{%
551       \endgroup%
552       \def\noexpand\filename@area{\filename@area}%
553       \def\noexpand\filename@base{\filename@base}%
554       \ifx\filename@ext\relax%
555         \let\noexpand\filename@ext\noexpand\relax%
556       \else%
557         \def\noexpand\filename@ext{\filename@ext}%
558       \fi%
559     }%
560   \svg@tempa%
561 }
```

`\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 occurence 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.

```
562 \newcommand*\svg@extension@parse[1]{%
563   \IfArgIsEmpty{#1}{}{%
564     \ifstr{#1}{\filename@ext}{}{%
565       \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.

```
566         \edef\svg@tempa{%
567           \def\noexpand\svg@tempa{}%
568           \let\noexpand\svg@tempb\relax%
569           \noexpand\svg@extension@@parse%
```

```
570        \filename@ext.\noexpand\@nil#1\noexpand\@nil%
571      }%
572      \svg@tempa%
573      \edef\svg@tempa{%
574        \endgroup%
```

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

```
575        \def\noexpand\filename@base{\filename@base\svg@tempa}%
576        \ifx\svg@tempb\relax%
577          \let\noexpand\filename@ext\relax%
578        \else%
579          \def\noexpand\filename@ext{\svg@tempb}%
580        \fi%
581      }%
582      \svg@tempa%
583    }%
584  }%
585 }
```

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

```
586 \newcommand*\svg@extension@@parse{}
587 \def\svg@extension@@parse#1.#2\@nil#3\@nil{%
588   \edef\svg@tempa{\svg@tempa.#1}%
589   \IfArgIsEmpty{#2}{}{%
590     \ifstr{#2}{#3.}{%
```

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

```
591        \edef\svg@tempb{#3}%
592      }{%
593        \svg@extension@@parse#2\@nil#3\@nil%
594      }%
595    }%
596 }
```

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

```
597 \newcommand*\svg@file@missing[3][]{%
598   \begingroup%
599     \svg@quotes@remove[{#2}]{\svg@tempa}%
600     \svg@filename@parse[{#1}]{\svg@tempa}%
601     \IfArgIsEmpty{#1}{%
602       \svg@quotes@remove[{#3}]{\svg@tempb}%
603       \def\svg@tempa{%
604         Did you run the export with Inkscape? There's no file\MessageBreak%
605         '\filename@area\filename@base.\filename@ext'\MessageBreak%
606         although '\svg@tempb' was found.%
607       }%
608     }{%
609       \edef\filename@ext{#1}%
610       \ifstr{\filename@area}{./}{\let\filename@area\@empty}{}%
```

Collecting all considered path for the error message.

```
611        \edef\svg@tempb{#3}%
612        \ifstr{\svg@tempb}{./}{\let\svg@tempb\@empty}{}%
613        \ifx\svg@tempb\@empty%
614          \svg@set@input@path%
615        \else%
616          \svg@set@input@path[\svg@tempb]%
617        \fi%
618        \ifx\input@path\@undefined%
```

```
619          \def\svg@tempb{No additional path was given.}%
620        \else%
621          \def\svg@tempb{Following folders have additionally been searched:}%
622          \expandafter\@tfor\expandafter\svg@tempa\expandafter:\expandafter=%
623              \input@path\do{%
624            \edef\svg@tempb{\svg@tempb\noexpand\MessageBreak\svg@tempa}%
625          }%
626        \fi%
```

The error message itself.

```
627        \def\svg@tempa{%
628          There's no file '\filename@base.\filename@ext'\MessageBreak%
629          \ifx\filename@area\@empty%
630            neither in the current directory nor any other searched\MessageBreak%
631            path given by \string\svgpath\space or \string\graphicspath.%
632            \MessageBreak\svg@tempb%
633          \else%
634            in folder '\filename@area'.%
635          \fi%
636        }%
637      }%
638      \PackageError{svg}{%
639        File '\filename@base.\filename@ext' is missing%
640      }{\svg@tempa}%
641    \endgroup%
642 }
```

\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 is
                    definied. Unfortunately this functionality isn't provided by XeTeX.

```
643 \ifx\pdf@filemoddate\@undefined
644   \newcommand*\svg@iffilenewer[2]{\@gobbletwo}
645 \else
646   \newcommand*\svg@iffilenewer[2]{%
647     \begingroup%
648       \edef\svg@tempa{\pdf@filemoddate{#1}}%
649       \edef\svg@tempb{\pdf@filemoddate{#2}}%
650       \ifnum\pdf@strcmp{\svg@tempa}{\svg@tempb}>\z@\relax%
651         \aftergroup\@firstoftwo%
652       \else%
653         \aftergroup\@secondoftwo%
654       \fi%
655     \endgroup%
656   }
657 \fi
```

B.3. Optional Parameters for user commands

\svg@local@param@set   Most of the package options can also be used as optional parameters for \includesvg or
\svg@local@param@use   \includeinkscape. Some of them are overloaded for the usage as optional argument and
\svg@local@param@def   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**.

```
658 \newcommand*\svg@local@param@set[1]{%
659   \svg@local@param@use%
660   \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.

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

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

```
662    \@svg@tempswafalse%
663    \ifdim\svg@param@width>\z@\relax\ifdim\svg@param@height>\z@\relax%
664      \@svg@tempswatrue%
665    \fi\fi%
666    \if@svg@tempswa\else%
667      \FamilyOptions{SVG}{distort=false}%
668    \fi%
669 }
670 \newcommand*\svg@local@param@use{}
671 \newcommand*\svg@local@param@def[1]{%
672    \edef\svg@local@param@use{%
673      \unexpanded\expandafter{\svg@local@param@use}\unexpanded{#1}%
674    }%
675 }
```

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

```
676 ⟨*body⟩
677 \DefineFamilyMember[.param]{SVG}
678 ⟨/body⟩
```

B.4. User commands

\svgsetup The macro `\svgsetup` can be used to change options after loading the package **svg** both in
\setsvg   preamble and the document body. For compatibility reasons, `\setsvg` is also defined.

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

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

```
681 \newcommand*\svg@input@path{}
682 \let\svg@input@path\input@path
683 \newcommand*\svgpath[1]{%
684    \def\svg@tempa##1\@nil{%
685      \ifx\svg@tempb\bgroup%
686        \def\svg@input@path{#1}%
687      \else%
688        \def\svg@input@path{{#1}}%
689      \fi%
690    }%
691    \futurelet\svg@tempb\svg@tempa#1\@nil%
692 }
```

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

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

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

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

inkscape (param.)   Most of the optional parameters have the same effect as the identically named options.
inkscapeformat (param.) Only parameter `lastpage` is extended (see below). Moreover, there are some additional
inkscapelatex (param.)   parameters, which can only be used as optional argument for `\includesvg` (`angle` and
inkscapearea (param.)   `origin`) but not as an option. Now all parameters are set in local context (within a group).
inkscapedpi (param.)
inkscapeopt (param.)   ```
svgextension (param.)   696   \svg@local@param@set{#1}%
width (param.)   ```
height (param.)
distort (param.)   31
scale (param.)
pretex (param.)
apptex (param.)
draft (param.)
```

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

```
697    \if@svg@ink@latex%
698      \edef\svg@file@suffix{_\svg@file@ext-tex}%
699    \else%
700      \edef\svg@file@suffix{_\svg@file@ext-raw}%
701    \fi%
702    \@onelevel@sanitize\svg@file@suffix%
```

Searching all given paths for the relevant SVG file.

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

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

```
705      \svg@ink@run%
706      \IfFileExists{\svg@out@base}{}{%
707        \@svg@file@foundfalse%
708        \svg@file@missing{\svg@out@base}{\svg@file@base.\svg@file@ext}%
709      }%
710      \if@svg@ink@latex%
711        \IfFileExists{\svg@out@base_tex}{}{%
712          \@svg@file@foundfalse%
713          \svg@file@missing{\svg@out@base_tex}{\svg@file@base.\svg@file@ext}%
714        }%
715      \fi%
```

Include the resulting graphic file and maybe extract independent files.

```
716      \if@svg@file@found%
717        \svg@input{\svg@out@base}%
718        \svg@extract{\svg@out@base}%
719      \fi%
720    \else%
```

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

```
721      \svg@file@missing[\svg@file@ext]{\svg@file@base}{}%
722    \fi%
723  \endgroup%
724 }
```

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

```
725 \svg@local@param@def{%
726   \FamilyCounterKey[.param]{SVG}{lastpage}{svg@param@lastpage}%
727 }
```

angle (param.)   The parameters `angle` and `origin` are definied as pendants to the keys provided by
origin (param.)   \includegraphics.

```
728 \newcommand*\svg@param@angle{0}
729 \svg@local@param@def{%
730   \DefineFamilyKey[.param]{SVG}{angle}{%
731     \ifisdimension{#1\p@}{%
732       \renewcommand*\svg@param@angle{#1}%
733       \FamilyKeyStateProcessed%
734     }{}%
735   }%
736 }
737 \newcommand*\svg@param@origin{c}
738 \svg@local@param@def{%
739   \DefineFamilyKey[.param]{SVG}{origin}[c]{%
```

32

```
740         \renewcommand*\svg@param@origin{#1}%
741         \FamilyKeyStateProcessed%
742     }%
743 }
```

\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.

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

Checking for deprecated commands \svgwidth and \svgscale.

```
746     \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.

```
747     \svg@filename@parse{#2}%
748     \ifx\filename@ext\relax\else%
749       \svg@quotes@remove{\filename@ext}%
750       \expandafter\lowercase\expandafter{%
751         \expandafter\def\expandafter\filename@ext\expandafter{\filename@ext}%
752       }%
753       \def\svg@tempb##1_tex##2\@nil{%
754         \IfArgIsEmpty{##1}{}{\def\filename@ext{##1}}%
755         \ifstr{##2}{_tex}{\@svg@tempswatrue}{\@svg@tempswafalse}%
756       }%
757       \@svg@tempswafalse%
758       \@tfor\svg@tempa:={pdf}{eps}{ps}{png}\do{%
759         \begingroup%
760           \expandafter\svg@tempb\filename@ext_tex\@nil%
761           \svg@extension@parse{\svg@tempa}%
762           \ifx\filename@ext\relax%
763             \def\svg@tempb{\endgroup}%
764           \else%
765             \edef\svg@tempb{%
766               \endgroup%
767               \noexpand\FamilyOptions{SVG}{inkscapeformat=\svg@tempa}%
768               \if@svg@tempswa%
769                 \noexpand\FamilyOptions{SVG}{inkscapelatex=true}%
770               \fi%
771               \def\noexpand\filename@base{\filename@base}%
772               \def\noexpand\filename@ext{\filename@ext}%
773               \noexpand\@svg@tempswatrue%
774             }%
775           \fi%
776         \svg@tempb%
```

Break for loop, if valid extension was found.

```
777         \if@svg@tempswa%
778           \@break@tfor%
779         \fi%
780       }%
```

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

```
781       \if@svg@tempswa\else%
782         \svg@extension@parse{\svg@ink@format}%
783       \fi%
784     \fi%
```

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).

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

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

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

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

Checking the required files for graphic inclusion.

```
788        \edef\svg@out@name{\svg@file@name}%
789        \edef\svg@out@base{\svg@file@path\svg@file@name.\svg@ink@format}%
790        \if@svg@ink@latex%
791            \IfFileExists{\svg@out@base_tex}{}{%
792                \@svg@file@foundfalse%
793                \svg@file@missing{\svg@out@base_tex}{\svg@out@base}%
794            }%
795        \fi%
```

Include the resulting graphic file and maybe extract independent files.

```
796        \if@svg@file@found%
797            \svg@input{\svg@out@base}%
798            \svg@extract{\svg@out@base}%
799        \fi%
800    \else%
```

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

```
801        \svg@file@missing[\svg@ink@format]{\svg@file@base}{\svg@out@path}%
802    \fi%
803    \endgroup%
804 }
```

B.5. Auxiliary macros

\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.

```
805 \newcommand*\svg@deprecated@param{%
806    \@svg@tempswafalse%
807    \ifx\svgwidth\@undefined\else%
808        \edef\svg@tempa{\noexpand\FamilyOptions{SVG}{width=\svgwidth}}%
809        \svg@tempa%
810        \@svg@tempswatrue%
811    \fi%
812    \ifx\svgscale\@undefined\else%
813        \edef\svg@tempa{\noexpand\FamilyOptions{SVG}{scale=\svgscale}}%
814        \svg@tempa%
815        \@svg@tempswatrue%
816    \fi%
817    \if@svg@tempswa%
818        \PackageWarning{svg}{%
819            You should specify the image size with parameters\MessageBreak%
820            'width' and 'height' or 'scale' instead of using\MessageBreak%
821            '\string\svgscale' or '\string\svgwidth'%
822        }%
823        \let\svgwidth\@undefined%
824        \let\svgscale\@undefined%
825    \fi%
826 }
```

<dl>
<dt>\svg@ink@run</dt>
<dt>\if@svg@ink@run</dt>
</dl>

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

```
827 \newif\if@svg@ink@run
828 \newcommand*\svg@ink@run{%
829   \ifnum\svg@ink@mode>\z@\relax%
830     \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.

```
831     \@svg@ink@runtrue%
832     \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.

```
833       \svg@iffilenewer{\svg@file@base.\svg@file@ext}{\svg@out@base}{}{%
834         \@svg@ink@runfalse%
835       }%
```

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.

```
836       \if@svg@ink@latex%
837         \IfFileExists{\svg@out@base_tex}{%
838           \ifnum\pdf@shellescape=\@ne\relax\if@svg@ink@run%
839             \svg@iffilenewer{\svg@out@base_tex}{\svg@out@base}{%
840               \@svg@ink@runfalse%
841               \svg@quotes@remove[\svg@out@base]{\svg@tempa}%
842               \PackageWarning{svg}{%
843                 Since the encountered filedate of file\MessageBreak%
844                 '\svg@tempa_tex' is newer than \MessageBreak%
845                 '\svg@tempa' it's supposed that\MessageBreak%
846                 you customized this file. To avoid an accidental\MessageBreak%
847                 overwriting of this file, the Inkscape export\MessageBreak%
848                 won't be done. If you want to overwrite the\MessageBreak%
849                 existing file please choose the parameter\MessageBreak%
850                 'inkscape=force'%
851               }%
852             }{}%
853           \fi\fi%
854         }{\@svg@ink@runtrue}%
855       \fi%
856     \fi%
```

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

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

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

```
859         \ifx\svg@ink@dpi\relax%
860           \ifstr{\svg@ink@format}{png}{%
861             \FamilyOptions{SVG}{inkscapedpi=300}%
862           }{}%
863         \fi%
864         \PackageInfo{svg}{%
865           Calling Inkscape%
866           \ifx\svg@ink@opt\@empty\else%
867             \space with added options '\svg@ink@opt'%
868           \fi%
869         }%
```

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

```
870          \svg@quotes@remove[\svg@file@base]{\svg@tempa}%
871          \svg@quotes@remove[\svg@out@name]{\svg@tempb}}%
872          \ShellEscape{\svg@ink@cmd{\svg@tempa}{\svg@tempb}}%
873          \IfFileExists{\svg@out@name.\svg@ink@format}{%
874            \edef\svg@tempb{\svg@tempb.\svg@ink@format}%
875            \svg@quotes@remove{\svg@out@base}%
876            \svg@shell@mkdir{\svg@out@path}%
877            \svg@shell@move{\svg@tempb}{\svg@out@base}%
878            \if@svg@ink@latex%
879              \svg@shell@move{\svg@tempb_tex}{\svg@out@base_tex}%
880            \fi%
881          }{%
882            \PackageWarning{svg}{%
883              The export with Inkscape failed for file\MessageBreak%
884              ‘\svg@tempa.\svg@file@ext’\MessageBreak%
885              Troubleshooting: Please check in the log file how\MessageBreak%
886              the invocation of Inkscape took place and try to\MessageBreak%
887              execute it yourself in the terminal%
888            }%
889          }%
```

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

```
890        \else%
891          \svg@quotes@remove[\svg@file@base]{\svg@tempa}%
892          \PackageWarning{svg}{%
893            You didn't enable ‘shell escape’ (or ‘write18’)\MessageBreak%
894            so it wasn't possible to launch the Inkscape export\MessageBreak%
895            for ‘\svg@tempa.\svg@file@ext’%
896          }%
897        \fi%
898      \fi%
899      \endgroup%
900    \fi%
901 }
```

\svg@ink@cmd   The actual call of **Inkscape** at command line.

```
902 \newcommand*\svg@ink@cmd[2]{%
903   \svg@ink@exe\space-z\space\svg@ink@area\space%
904   \ifx\svg@ink@dpi\relax\else--export-dpi=\svg@ink@dpi\space\fi%
905   \if@svg@ink@latex--export-latex\space\fi%
906   \svg@ink@opt\space%
907   --file="#1.\svg@file@ext"\space%
908   --export-\svg@ink@format="#2.\svg@ink@format"\space%
909 }
```

\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.

```
910 \newcommand*\svg@get@lastpage[1]{%
911   \ifstr{\svg@ink@format}{pdf}{%
912     \begingroup%
913       \@tempcnta=\m@ne\relax%
914       \ifx\XeTeXpdfpagecount\@undefined%
915         \ifpdf%
916           \ifx\pdfximage\@undefined%
917             \ifx\saveimageresource\@undefined\else%
918               \saveimageresource{#1}%
919               \@tempcnta=\lastsavedimageresourcepages\relax%
920             \fi%
```

```
921        \else%
922          \pdfximage{#1}%
923          \@tempcnta=\pdflastximagepages\relax%
924        \fi%
925      \fi%
926    \else%
927      \@tempcnta=\XeTeXpdfpagecount#1\relax%
928    \fi%
929    \ifnum\@tempcnta=\m@ne\relax%
930      \PackageWarning{svg}{%
931        It wasn't possible to detect the last page\MessageBreak%
932        of '#1'%
933      }%
934    \else%
935      \PackageInfo{svg}{Last page of '#1' is \the\@tempcnta}%
936    \fi%
937    \edef\svg@tempa{%
938      \endgroup%
939      \noexpand\FamilyOptions{SVG}{lastpage=\the\@tempcnta}%
940    }%
941    \svg@tempa%
942  }{}%
943 }
```

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

```
944 \newcommand*\svg@wrn@scale{%
945   \ifdim\dimexpr\svg@param@scale\p@\relax=\p@\relax\else%
946     \@svg@tempswafalse%
947     \ifdim\svg@param@width>\z@\relax%
948       \@svg@tempswatrue%
949     \fi%
950     \ifdim\svg@param@height>\z@\relax%
951       \@svg@tempswatrue%
952     \fi%
953     \if@svg@tempswa%
954       \PackageWarning{svg}{%
955         The parameter 'scale' is only considered if neither\MessageBreak%
956         'width' nor 'height' are specified%
957       }%
958     \fi%
959   \fi%
960 }
```

\svg@input   With \svg@@input the export results of **Inkscape** are included. The macro \svg@input
\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.

```
961 \newcommand*\svg@input{\svg@@input}
962 \newcommand*\svg@@input[2][]{%
963   \IfArgIsEmpty{#1}{}{\svg@local@param@set{#1}}%
964   \svg@set@input@path%
965   \if@svg@draft%
966     \@svg@ink@latexfalse%
967   \fi%
```

In order to support file names with multiple dots, the second argument is parsed and only
the part after the last dot is stroed in \svg@tempb as extension. Everything before is stored
in \svg@tempa.

```
968   \def\svg@tempb##1.##2\@nil{%
969     \IfArgIsEmpty{##2}{%
970       \def\svg@tempb{##1}%
```

37

```
971    }{%
972      \edef\svg@tempa{\svg@tempa.##1}%
973      \svg@tempb##2\@nil%
974    }%
975  }%
976  \edef\svg@tempa{%
977    \def\noexpand\svg@tempa{}%
978    \noexpand\svg@tempb#2.\noexpand\@nil%
979  }%
980  \svg@tempa%
```

Afterwards `\svg@tempa` is defined with the file name itself within enclosing braces followed by the extension and `\svg@tempb` holds the original file name plus extension without enclosing braces.

```
981  \svg@remove@leadingchar.\svg@tempa%
982  \edef\svg@tempa{{\svg@tempa}.\svg@tempb}%
983  \edef\svg@tempb{#2}%
```

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`.

```
984  \if@svg@ink@latex%
985    \svg@patches{\svg@tempa}%
986    \ifnum\value{svg@param@lastpage}=\z@\relax%
987      \expandafter\svg@get@lastpage\expandafter{\svg@tempb}%
988    \fi%
989    \edef\svg@tempa{%
990      \ifx\svg@param@pretex\relax\else%
991        \noexpand\svg@param@pretex%
992      \fi%
993      \noexpand\input{\svg@tempb_tex}%
994      \ifx\svg@param@apptex\relax\else%
995        \noexpand\svg@param@apptex%
996      \fi%
997    }%
```

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

```
998    \if@svg@param@distort%
999      \def\svg@tempb{\resizebox*{\svg@param@width}{\svg@param@height}}%
1000    \else%
1001      \let\svg@tempb\@firstofone%
1002    \fi%
1003    \sbox\svg@box{\svg@tempb{\svg@tempa}}%
```

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

```
1004    \ifdim\dimexpr\svg@param@angle\p@\relax=\z@\relax%
1005      \let\svg@tempb\@firstofone%
1006    \else%
1007      \edef\svg@tempb{%
1008        \noexpand\rotatebox[origin=\svg@param@origin]{\svg@param@angle}%
1009      }%
1010    \fi%
1011    \svg@tempb{\usebox\svg@box}%
1012  \else%
```

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

```
1013    \svg@wrn@scale%
1014    \edef\svg@tempb{%
1015      draft\if@svg@draft\else=false\fi,%
1016      scale=\svg@param@scale,%
1017      keepaspectratio\if@svg@param@distort=false\fi%
```

```
1018      }%
1019      \ifdim\svg@param@height>\z@\relax%
1020        \edef\svg@tempb{\svg@tempb,height=\svg@param@height}%
1021      \fi%
1022      \ifdim\svg@param@width>\z@\relax%
1023        \edef\svg@tempb{\svg@tempb,width=\svg@param@width}%
1024      \fi%
1025      \ifdim\dimexpr\svg@param@angle\p@\relax=\z@\relax\else%
1026        \edef\svg@tempb{%
1027          \svg@tempb,origin=\svg@param@origin,angle=\svg@param@angle%
1028        }%
1029      \fi%
1030      \expandafter\includegraphics\expandafter[\svg@tempb]{\svg@tempa}%
1031    \fi%
1032 }
```

B.6. Patches

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

```
1033 \newcommand*\svg@picture@saved{}
1034 \let\svg@picture@saved\picture
1035 \newcommand*\svg@includegraphics@saved{}
1036 \let\svg@includegraphics@saved\includegraphics
1037 \newcommand*\svg@patches[1]{%
1038    \let\picture\svg@picture@patched%
1039    \let\includegraphics\svg@includegraphics@patched%
1040    \edef\svg@includegraphics@file{#1}%
1041 }
```

\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.

```
1042 \newcommand*\svg@picture@patched{}
1043 \newcommand*\svg@pictur@patched{}
1044 \long\def\svg@picture@patched#1{\svg@pictur@patched@#1}
1045 \def\svg@pictur@patched@(#1,#2){%
1046    \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.

```
1047    \ifdim\svg@param@height>\z@\relax%
1048      \Gscale@div\svg@tempa{#1\p@}{#2\p@}%
1049      \setlength\unitlength{\svg@param@height}%
1050      \setlength\unitlength{\svg@tempa\unitlength}%
1051      \ifdim\svg@param@width>\z@\relax%
1052        \ifdim\unitlength>\svg@param@width\relax%
1053          \setlength\unitlength{\svg@param@width}%
1054        \fi%
1055      \fi%
1056    \else%
```

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

```
1057      \ifdim\svg@param@width>\z@\relax%
1058        \setlength\unitlength{\svg@param@width}%
1059      \else%
1060        \setlength\unitlength{\svg@param@scale\unitlength}%
1061      \fi%
1062    \fi%
```

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

```
1063    \svg@picture@saved(#1,#2)%
1064 }
```

<table>
<tr><td>\svg@includegraphics@patched</td></tr>
<tr><td>\svg@includegraphics@file</td></tr>
</table>

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.

```
1065 \DefineFamily{SVGpatch}
1066 \DefineFamilyMember{SVGpatch}
1067 \newcounter{svg@param@currpage}
1068 \setcounter{svg@param@currpage}{\m@ne}
1069 \FamilyCounterKey{SVGpatch}{page}{svg@param@currpage}
1070 \DefineFamilyKey{SVGpatch}{width}{\FamilyKeyStateProcessed}
1071 \newcommand*\svg@includegraphics@file{}
1072 \newcommand*\svg@includegraphics@patched[2][]{%
1073    \FamilyOptions{SVGpatch}{#1}%
```

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

```
1074    \ifnum\value{svg@param@lastpage}<\z@\relax%
1075      \FamilySetCounter{SVGpatch}{page}{svg@param@currpage}{%
1076        \the\value{svg@param@lastpage}%
1077      }%
1078    \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`.

```
1079    \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.

```
1080      \svg@includegraphics@saved[{#1}]{\svg@includegraphics@file}%
1081    \fi%
1082 }
```

C. Extracting independent graphic files with svg-extract

C.1. Options

For package **svg-extract** the user interface is extended. The following options can either be set with `\svgsetup` or be used as local 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.

```
1083 ⟨*base⟩
1084 \DefineFamilyMember[.dummy]{SVG}
1085 \newcommand*\svg@dummy@key[2][]{%
1086    \@ifpackageloaded{svg-extract}{}{%
1087      \IfArgIsEmpty{#1}{%
1088        \DefineFamilyKey[.dummy]{SVG}{#2}{%
1089          \PackageWarning{svg}{%
1090            The option key '#2' can only\MessageBreak%
```

```
1091        be used with package 'svg-extract', but\MessageBreak%
1092          you didn't load it%
1093        }%
1094        \FamilyKeyStateProcessed%
1095      }%
1096    }{%
1097      \DefineFamilyKey[.dummy]{SVG}{#2}[{#1}]{%
1098        \PackageWarning{svg}{%
1099          The option key '#2' can only\MessageBreak%
1100          be used with package 'svg-extract', but\MessageBreak%
1101          you didn't load it%
1102        }%
1103        \FamilyKeyStateProcessed%
1104      }%
1105    }%
```

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

```
1106      \AfterPackage{svg-extract}{\RelaxFamilyKey[.dummy]{SVG}{#2}}%
1107    }%
1108 }
1109 ⟨/base⟩
```

C.1.1. Controlling the extract process

extract (opt.)   With option `extract` it can be controlled, if the extraction of independent graphic files
\if@svgx@run   should be done.

```
1110 ⟨∗base⟩
1111 \svg@dummy@key[true]{extract}
1112 ⟨/base⟩
1113 ⟨∗extract⟩
1114 \newif\if@svgx@run
1115 \DefineFamilyKey{SVG}{extract}[true]{%
1116   \lowercase{\def\svg@tempa{#1}}%
1117   \FamilySetNumerical{SVG}{extract}{svg@tempa}{%
1118     {false}{0},{off}{0},{no}{0},%
1119     {true}{1},{on}{1},{yes}{1},{onlynewer}{1},{newer}{1},%
1120     {overwrite}{1},{force}{1},{forced}{1},%
1121     {pdf}{2},{eps}{3},{ps}{4}%
1122   }{\svg@tempa}%
1123   \ifx\FamilyKeyState\FamilyKeyStateProcessed%
1124     \ifcase\svg@tempa\relax% false
1125       \@svgx@runfalse%
1126     \or% true
1127       \@svgx@runtrue%
1128     \or% pdf
1129       \FamilyOptions{SVG}{extractformat=pdf}%
1130     \or% eps
1131       \FamilyOptions{SVG}{extractformat=eps}%
1132     \or% ps
1133       \FamilyOptions{SVG}{extractformat=ps}%
1134     \fi%
1135   \fi%
1136 }
1137 ⟨/extract⟩
```

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

```
1138 ⟨∗extract⟩
1139 \DeclareOption{on}{\FamilyOptions{SVG}{extract=true}}
1140 \DeclareOption{off}{\FamilyOptions{SVG}{extract=false}}
1141 ⟨/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.

```
1142 ⟨∗base⟩
1143 \svg@dummy@key{extractformat}
1144 \svg@dummy@key[true]{pdf}
1145 \svg@dummy@key[true]{eps}
1146 ⟨/base⟩
1147 ⟨∗extract⟩
1148 \newcommand*\svgx@format{pdf}
1149 \ifxetex\else\ifpdf\else
1150   \renewcommand*\svgx@format{eps}
1151 \fi\fi
1152 \DefineFamilyKey{SVG}{extractformat}{%
1153   \lowercase{\edef\svgx@format{#1}}%
1154   \FamilyKeyStateProcessed%
1155 }
1156 \DefineFamilyKey{SVG}{pdf}[true]{%
1157   \FamilySetBool{SVG}{pdf}{@svg@tempswa}{#1}%
1158   \ifx\FamilyKeyState\FamilyKeyStateProcessed%
1159     \if@svg@tempswa%
1160       \svgx@ifinlist{pdf}{\svgx@format}{}{%
1161         \edef\svgx@format{\svgx@format,pdf}%
1162       }%
1163       \svg@deprecated@key{pdf}{extractformat={\svgx@format}}%
1164     \else%
1165       \FamilyKeyStateUnknownValue%
1166     \fi%
1167   \fi%
1168 }
1169 \DefineFamilyKey{SVG}{eps}[true]{%
1170   \FamilySetBool{SVG}{eps}{@svg@tempswa}{#1}%
1171   \ifx\FamilyKeyState\FamilyKeyStateProcessed%
1172     \if@svg@tempswa%
1173       \svgx@ifinlist{eps}{\svgx@format}{}{%
1174         \edef\svgx@format{\svgx@format,eps}%
1175       }%
1176       \svg@deprecated@key{eps}{extractformat={\svgx@format}}%
1177     \else%
1178       \FamilyKeyStateUnknownValue%
1179     \fi%
1180   \fi%
1181 }
1182 ⟨/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}`.

```
1183 ⟨∗base⟩
1184 \svg@dummy@key{extractpreamble}
1185 \svg@dummy@key{preamble}
1186 \svg@dummy@key{extractpreambleend}
1187 \svg@dummy@key{end}
1188 ⟨/base⟩
1189 ⟨∗extract⟩
1190 \newcommand*\svgx@preamble{\jobname.\svgx@latex@ext}%
1191 \DefineFamilyKey{SVG}{extractpreamble}{%
1192   \renewcommand*\svgx@preamble{#1}%
1193   \FamilyKeyStateProcessed%
1194 }
1195 \DefineFamilyKey{SVG}{preamble}{%
1196   \svg@deprecated@key[svg-extract]{preamble=#1}{extractpreamble=#1}%
1197 }
1198 \newcommand*\svgx@endpreamble{}
```

```
1199 \expandafter\def\expandafter\svgx@endpreamble\expandafter{%
1200   \csname begin\endcsname{document}%
1201 }
1202 \DefineFamilyKey{SVG}{extractpreambleend}{%
1203   \renewcommand*\svgx@endpreamble{#1}%
1204   \FamilyKeyStateProcessed%
1205 }
1206 \DefineFamilyKey{SVG}{end}{%
1207   \svg@deprecated@key[svg-extract]{end=#1}{extractpreambleend=#1}%
1208 }
1209 ⟨/extract⟩
```

extractruns (opt.)   With this option, the number of LaTeX runs for the separate auxiliary file can be set.
svgx@runs (counter)

```
1210 ⟨*base⟩
1211 \svg@dummy@key{extractruns}
1212 ⟨/base⟩
1213 ⟨*extract⟩
1214 \newcounter{svgx@runs}
1215 \DefineFamilyKey{SVG}{extractruns}{%
1216   \FamilySetCounter{SVG}{extractruns}{svgx@runs}{#1}%
1217   \ifx\FamilyKeyState\FamilyKeyStateProcessed%
1218     \ifnum\value{svgx@runs}<\@ne\relax%
1219       \PackageWarning{svg-extract}{%
1220         The count for runs has to be at least one%
1221       }%
1222       \FamilySetCounter{SVG}{extractruns}{svgx@runs}{\@ne}%
1223     \fi%
1224   \fi%
1225 }
1226 ⟨/extract⟩
```

latexexe (opt.)   The command and facultative options for the LaTeX call of the separate auxiliary file. The
pdflatex (opt.)   default is set according to the currently used compiler.
\svgx@latex@exe
latexext (opt.)
\svgx@latex@ext
latexopt (opt.)
\svgx@latex@opt

```
1227 ⟨*base⟩
1228 \svg@dummy@key{latexexe}
1229 \svg@dummy@key{pdflatex}
1230 \svg@dummy@key{latexext}
1231 \svg@dummy@key{latexopt}
1232 ⟨/base⟩
1233 ⟨*extract⟩
1234 \ifxetex
1235   \newcommand*\svgx@latex@exe{xelatex}
1236 \else\ifluatex
1237   \ifpdf
1238     \newcommand*\svgx@latex@exe{lualatex}
1239   \else
1240     \newcommand*\svgx@latex@exe{lualatex --output-format=dvi}
1241   \fi
1242 \else\ifpdf
1243   \newcommand*\svgx@latex@exe{pdflatex}
1244 \else
1245   \newcommand*\svgx@latex@exe{latex}
1246 \fi\fi\fi
1247 \DefineFamilyKey{SVG}{latexexe}{%
1248   \renewcommand*\svgx@latex@exe{#1}%
1249   \FamilyKeyStateProcessed%
1250 }
1251 \DefineFamilyKey{SVG}{pdflatex}{%
1252   \svg@deprecated@key[svg-extract]{pdflatex=#1}{latexexe=#1}%
1253 }
1254 \newcommand*\svgx@latex@ext{tex}
1255 \DefineFamilyKey{SVG}{latexext}{%
1256   \renewcommand*\svgx@latex@ext{#1}%
```

43

```
1257    \FamilyKeyStateProcessed%
1258 }
1259 \newcommand*\svgx@latex@opt{}
1260 \DefineFamilyKey{SVG}{latexopt}{%
1261    \renewcommand*\svgx@latex@opt{#1}%
1262    \FamilyKeyStateProcessed%
1263 }
1264 ⟨/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 2_ε distributions. These are used to generate all files, which are supported by option extractformat, as they don't need an additional application.

```
1265 ⟨*base⟩
1266 \svg@dummy@key{dvipsopt}
1267 \svg@dummy@key{pstoepsopt}
1268 \svg@dummy@key{pstopdfopt}
1269 \svg@dummy@key{pdftoepsopt}
1270 \svg@dummy@key{pdftopsopt}
1271 \svg@dummy@key{pdftops}
1272 ⟨/base⟩
1273 ⟨*extract⟩
1274 \newcommand*\svgx@dvips@exe{dvips}
1275 \newcommand*\svgx@dvips@opt{}
1276 \DefineFamilyKey{SVG}{dvipsopt}{%
1277    \renewcommand*\svgx@dvips@opt{#1}%
1278    \FamilyKeyStateProcessed%
1279 }
1280 \newcommand*\svgx@pstoeps@exe{ps2eps}
1281 \newcommand*\svgx@pstoeps@opt{-B -C}
1282 \DefineFamilyKey{SVG}{pstoepsopt}{%
1283    \renewcommand*\svgx@pstoeps@opt{#1}%
1284    \FamilyKeyStateProcessed%
1285 }
1286 \newcommand*\svgx@pstopdf@exe{ps2pdf}
1287 \newcommand*\svgx@pstopdf@opt{}
1288 \DefineFamilyKey{SVG}{pstopdfopt}{%
1289    \renewcommand*\svgx@pstopdf@opt{#1}%
1290    \FamilyKeyStateProcessed%
1291 }
1292 \newcommand*\svgx@pdftoeps@exe{pdftops -eps}
1293 \newcommand*\svgx@pdftoeps@opt{}
1294 \DefineFamilyKey{SVG}{pdftoepsopt}{%
1295    \renewcommand*\svgx@pdftoeps@opt{#1}%
1296    \FamilyKeyStateProcessed%
1297 }
1298 \newcommand*\svgx@pdftops@exe{pdftops}
1299 \newcommand*\svgx@pdftops@opt{}
1300 \DefineFamilyKey{SVG}{pdftopsopt}{%
1301    \renewcommand*\svgx@pdftops@opt{#1}%
1302    \FamilyKeyStateProcessed%
1303 }
1304 \DefineFamilyKey{SVG}{pdftops}{%
1305    \PackageWarning{#1}{%
1306       The option key 'pdftops' is deprecated.\MessageBreak%
1307       You should use either 'pdftoepsopt' or\MessageBreak%
1308       'pdftopsopt' instead. See the manual for\MessageBreak%
1309       more. Nothing was done%
1310    }%
1311    \FamilyKeyStateProcessed%
1312 }
1313 ⟨/extract⟩
```

C.1.2. Invoking external application for graphic conversion

Besides the use of a conversion tool supplied by LaTeX 2_ε, the applications **ImageMagick** and **Ghostscript** can be used for converting graphics.

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

```
1314 ⟨*base⟩
1315 \svg@dummy@key[true]{convert}
1316 ⟨/base⟩
1317 ⟨*extract⟩
1318 \newif\if@svgx@cnv@run
1319 \newcommand*\svgx@cnv@cmd{}
1320 \DefineFamilyKey{SVG}{convert}[true]{%
1321   \FamilySetNumerical{SVG}{convert}{svg@tempa}{%
1322     {false}{0},{off}{0},{no}{0},%
1323     {true}{1},{on}{1},{yes}{1},{onlynewer}{1},{newer}{1},%
1324     {overwrite}{1},{force}{1},{forced}{1},%
1325     {magick}{2},{imagemagick}{2},{convert}{2},%
1326     {gs}{3},{ghostscript}{3},%
1327     {gs64}{4},{ghostscript64}{4},%
1328     {gs32}{5},{ghostscript32}{5}%
1329   }{#1}%
1330   \ifx\FamilyKeyState\FamilyKeyStateProcessed%
1331     \ifcase\svg@tempa\relax% false
1332       \@svgx@cnv@runfalse%
1333     \or% true
1334       \@svgx@cnv@runtrue%
1335     \or% magick
1336       \@svgx@cnv@runtrue%
1337       \renewcommand*\svgx@cnv@cmd{\svgx@magick@cmd}%
1338     \or% gs
1339       \@svgx@cnv@runtrue%
1340       \renewcommand*\svgx@cnv@cmd{\svgx@gs@cmd}%
1341     \or% gs64
1342       \@svgx@cnv@runtrue%
1343       \renewcommand*\svgx@cnv@cmd{\svgx@gs@cmd}%
1344       \svgx@onlywindows{%
1345         \renewcommand*\svgx@gs@exe{gswin64c}%
1346       }%
1347     \or% gs32
1348       \@svgx@cnv@runtrue%
1349       \renewcommand*\svgx@cnv@cmd{\svgx@gs@cmd}%
1350       \svgx@onlywindows{%
1351         \renewcommand*\svgx@gs@exe{gswin32c}%
1352       }%
1353     \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.

```
1354     \else%
```

Same doing like with option `inkscape`.

```
1355       \def\svg@tempa##1-##2\@nil{%
1356         \IfArgIsEmpty{##2}{\def\svg@tempb{}}{%
1357           \def\svg@tempa##1####1\@nil{\def\svg@tempb{####1}}%
1358           \svg@tempa#1\@nil%
1359         }%
1360         \def\svg@tempa{##1}%
1361       }%
1362       \svg@tempa#1-\@nil%
```

```
1363    \PackageWarning{svg-extract}{%
1364      Setting the executable%
1365      \ifx\svg@tempb\@empty\else%
1366        \space and associated options%
1367      \fi%
1368      \MessageBreak%
1369      for ImageMagick should be done with options\MessageBreak%
1370      'magickexe=\svg@tempa'%
1371      \ifx\svg@tempb\@empty\else%
1372        \MessageBreak and 'magicksetting' and/or 'magickoperator'%
1373      \fi.\MessageBreak%
1374      Nevertheless, this was done by now%
1375      \ifx\svg@tempb\@empty\else%
1376        , whereby \MessageBreak 'magicksetting=\svg@tempb' was used%
1377      \fi%
1378    }%
1379    \FamilyOptions{SVG}{convert=magick}%
1380    \edef\svg@tempa{%
1381      \noexpand\FamilyOptions{SVG}{magickexe=\svg@tempa}%
1382      \ifx\svg@tempb\@empty\else%
1383        \noexpand\FamilyOptions{SVG}{magicksetting=\svg@tempb}%
1384      \fi%
1385    }%
1386    \svg@tempa%
1387  \fi%
1388 }
1389 ⟨/extract⟩
```

<div style="margin-left:2em">

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

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

</div>

```
1390 ⟨*base⟩
1391 \svg@dummy@key{convertformat}
1392 \svg@dummy@key[true]{png}
1393 ⟨/base⟩
1394 ⟨*extract⟩
1395 \newcommand*\svgx@cnv@format{png}
1396 \DefineFamilyKey{SVG}{convertformat}{%
1397   \lowercase{\edef\svgx@cnv@format{#1}}%
1398   \ifx\svgx@cnv@format\@empty\else%
1399     \@svgx@cnv@runtrue%
1400   \fi%
1401   \FamilyKeyStateProcessed%
1402 }
1403 \DefineFamilyKey{SVG}{png}[true]{%
1404   \FamilySetBool{SVG}{png}{@svg@tempswa}{#1}%
1405   \ifx\FamilyKeyState\FamilyKeyStateProcessed%
1406     \if@svg@tempswa%
1407       \svgx@ifinlist{png}{\svgx@cnv@format}{}{%
1408         \edef\svgx@cnv@format{\svgx@cnv@format,png}%
1409       }%
1410       \svg@deprecated@key{png}{convertformat={\svgx@cnv@format}}%
1411     \else%
1412       \FamilyKeyStateUnknownValue%
1413     \fi%
1414   \fi%
1415 }
1416 ⟨/extract⟩
```

<div style="margin-left:2em">

convertdpi (opt.)
convertdensity (opt.)
\svgx@cnv@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 recocnized by `\svgx@ifkeyandval`.

</div>

```
1417 ⟨∗base⟩
1418 \svg@dummy@key{convertdpi}
1419 \svg@dummy@key{convertdensity}
1420 ⟨/base⟩
1421 ⟨∗extract⟩
1422 \newcommand*\svgx@cnv@dpi{}
1423 \let\svgx@cnv@dpi\relax
1424 \DefineFamilyKey{SVG}{convertdpi}{%
1425   \FamilyKeyStateUnknownValue%
1426   \svgx@ifkeyandval{#1}{%
1427     \svgx@cnv@get@dpi{##2}%
1428     \ifx\svg@tempa\relax\else%
1429       \expandafter\edef\csname svgx@cnv@dpi@##1\endcsname{\svg@tempa}%
1430       \FamilyKeyStateProcessed%
1431     \fi%
1432   }{%
1433     \svgx@cnv@get@dpi{##1}%
1434     \ifx\svg@tempa\relax\else%
1435       \edef\svgx@cnv@dpi{\svg@tempa}%
1436       \FamilyKeyStateProcessed%
1437     \fi%
1438   }%
1439 }
1440 \DefineFamilyKey{SVG}{convertdensity}{\FamilyOptions{SVG}{convertdpi=#1}}
1441 ⟨/extract⟩
```

<div style="text-align:right">

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

</div>

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.

```
1442 ⟨∗base⟩
1443 \svg@dummy@key{magickexe}
1444 \svg@dummy@key{magicksetting}
1445 \svg@dummy@key{magickoperator}
1446 ⟨/base⟩
1447 ⟨∗extract⟩
1448 \newcommand*\svgx@magick@exe{}
1449 \DefineFamilyKey{SVG}{magickexe}{%
1450   \renewcommand*\svgx@magick@exe{#1}%
1451   \FamilyKeyStateProcessed%
1452 }
1453 \newcommand*\svgx@magick@set{}
1454 \DefineFamilyKey{SVG}{magicksetting}{%
1455   \svgx@setformatkey{#1}{svgx@magick@set}%
1456   \FamilyKeyStateProcessed%
1457 }
1458 \newcommand*\svgx@magick@opr{}
1459 \DefineFamilyKey{SVG}{magickoperator}{%
1460   \svgx@setformatkey{#1}{svgx@magick@opr}%
1461   \FamilyKeyStateProcessed%
1462 }
1463 ⟨/extract⟩
```

<div style="text-align:right">

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

</div>

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.

```
1464 ⟨∗base⟩
1465 \svg@dummy@key{gsexe}
1466 \svg@dummy@key{gsopt}
1467 \svg@dummy@key{gsdevice}
1468 ⟨/base⟩
1469 ⟨∗extract⟩
```

```
1470 \newcommand*\svgx@gs@exe{}
1471 \DefineFamilyKey{SVG}{gsexe}{%
1472   \renewcommand*\svgx@gs@exe{#1}%
1473   \FamilyKeyStateProcessed%
1474 }
1475 \newcommand*\svgx@gs@opt{}
1476 \DefineFamilyKey{SVG}{gsopt}{%
1477   \svgx@setformatkey{#1}{svgx@gs@opt}%
1478   \FamilyKeyStateProcessed%
1479 }
1480 \newcommand*\svgx@gs@device{}
1481 \DefineFamilyKey{SVG}{gsdevice}{%
1482   \svgx@setformatkey{#1}{svgx@gs@device}%
1483   \FamilyKeyStateProcessed%
1484 }
1485 ⟨/extract⟩
```

C.1.3. Setting output folder

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

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

```
1486 ⟨*base⟩
1487 \svg@dummy@key{extractpath}
1488 \svg@dummy@key{path}
1489 \svg@dummy@key{extractname}
1490 \svg@dummy@key{name}
1491 ⟨/base⟩
1492 ⟨*extract⟩
1493 \newcommand*\svgx@out@path{}
1494 \DefineFamilyKey{SVG}{extractpath}{%
1495   \svg@sanitize@dq\svg@tempb{#1}%
1496   \FamilySetNumerical{SVG}{extractpath}{svg@tempa}{%
1497     {svgpath}{0},{svgdir}{0},%
1498     {svgsubpath}{1},{svgsubdir}{1},%
1499     {basepath}{2},{basedir}{2},{jobpath}{2},{jobdir}{2},%
1500     {basesubpath}{3},{basesubdir}{3},{jobsubpath}{3},{jobsubdir}{3}%
1501   }{\svg@tempb}%
1502   \ifx\FamilyKeyState\FamilyKeyStateProcessed%
1503     \ifcase\svg@tempa\relax% svgpath
1504       \renewcommand*\svgx@out@path{\svg@file@path}%
1505     \or% svgsubpath
1506       \renewcommand*\svgx@out@path{\svg@file@path svg-extract/}%
1507     \or% basepath
1508       \renewcommand*\svgx@out@path{./}%
1509     \or% basesubpath
1510       \renewcommand*\svgx@out@path{./svg-extract/}%
1511     \fi%
1512   \else%
1513     \edef\svgx@out@path{\svg@tempb}%
1514     \svg@normalize@path{\svgx@out@path}%
1515     \FamilyKeyStateProcessed%
1516   \fi%
1517 }
1518 \DefineFamilyKey{SVG}{path}{%
1519   \svg@deprecated@key[svg-extract]{path=#1}{extractpath=#1}%
1520 }
1521 \newcounter{svgx@out@count}
1522 \newcommand*\svgx@out@name{}
1523 \newif\if@svgx@out@sec
1524 \DefineFamilyKey{SVG}{extractname}{%
1525   \svg@quotes@remove[{#1}]{\svg@tempb}%
1526   \FamilySetNumerical{SVG}{extractname}{svg@tempa}{%
```

```
1527     {filename}{0},{name}{0},%
1528     {filenamenumbered}{1},{namenumbered}{1},%
1529     {numberedfilename}{1},{numberedname}{1},%
1530     {numbered}{2},{section}{2},{numberedsection}{2},{sectionnumbered}{2}%
1531 }{\svg@tempb}%
1532 \@svgx@out@secfalse%
1533 \ifx\FamilyKeyState\FamilyKeyStateProcessed%
1534   \ifcase\svg@tempa\relax% filename
1535     \renewcommand*\svgx@out@name{\svg@out@name-extract}%
1536   \or% filenamenumbered
1537     \renewcommand*\svgx@out@name{\the\value{svgx@out@count}-\svg@out@name}%
1538   \or% numbered
1539     \renewcommand*\svgx@out@name{\the\value{svgx@out@count}-\svgx@out@sec}%
1540     \@svgx@out@sectrue%
1541   \fi%
1542 \else%
1543   \if@svg@quotes@found%
1544     \edef\svgx@out@name{"\svg@tempb"}%
1545   \else%
1546     \edef\svgx@out@name{\svg@tempb}%
1547   \fi%
1548   \FamilyKeyStateProcessed%
1549 \fi%
1550 }
1551 \DefineFamilyKey{SVG}{name}{%
1552   \svg@deprecated@key[svg-extract]{name=#1}{extractname=#1}%
1553 }
1554 ⟨/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.

```
1555 ⟨*base⟩
1556 \svg@dummy@key{extractwidth}
1557 \svg@dummy@key{extractheight}
1558 \svg@dummy@key{extractdistort}
1559 \svg@dummy@key{extractkeepaspectratio}
1560 \svg@dummy@key{extractscale}
1561 ⟨/base⟩
1562 ⟨*extract⟩
1563 \newcommand*\svgx@param@width{\svg@param@width}
1564 \DefineFamilyKey{SVG}{extractwidth}{%
1565   \FamilyKeyStateUnknownValue%
1566   \svg@ifvalueisrelax{#1}{%
1567     \renewcommand*\svgx@param@width{\z@}%
1568     \FamilyKeyStateProcessed%
1569   }{%
1570     \ifstr{#1}{inherit}{%
1571       \renewcommand*\svgx@param@width{\svg@param@width}%
1572       \FamilyKeyStateProcessed%
1573     }{%
1574       \FamilySetLengthMacro{SVG}{extractwidth}{\svgx@param@width}{#1}%
1575       \ifx\FamilyKeyState\FamilyKeyStateProcessed%
1576         \ifdim\svgx@param@width<\z@\relax%
1577           \FamilyKeyStateUnknownValue%
1578         \fi%
1579       \fi%
1580     }%
1581   }%
1582 }
```

```
1583 \newcommand*\svgx@param@height{\svg@param@height}
1584 \DefineFamilyKey{SVG}{extractheight}{%
1585   \FamilyKeyStateUnknownValue%
1586   \svg@ifvalueisrelax{#1}{%
1587     \renewcommand*\svgx@param@height{\z@}%
1588     \FamilyKeyStateProcessed%
1589   }{%
1590     \ifstr{#1}{inherit}{%
1591       \renewcommand*\svgx@param@height{\svg@param@height}%
1592       \FamilyKeyStateProcessed%
1593     }{%
1594       \FamilySetLengthMacro{SVG}{extractheight}{\svgx@param@height}{#1}%
1595       \ifx\FamilyKeyState\FamilyKeyStateProcessed%
1596         \ifdim\svgx@param@height<\z@\relax%
1597           \FamilyKeyStateUnknownValue%
1598         \fi%
1599       \fi%
1600     }%
1601   }%
1602 }
1603 \newif\if@svgx@param@distort
1604 \DefineFamilyKey{SVG}{extractdistort}[true]{%
1605   \FamilyKeyStateUnknownValue%
1606   \svg@ifvalueisrelax{#1}{%
1607     \@svgx@param@distortfalse%
1608     \FamilyKeyStateProcessed%
1609   }{%
1610     \ifstr{#1}{inherit}{%
1611       \renewcommand*\if@svgx@param@distort{\if@svg@param@distort}%
1612       \FamilyKeyStateProcessed%
1613     }{%
1614       \FamilySetBool{SVG}{extractdistort}{@svgx@param@distort}{#1}%
1615     }%
1616   }%
1617 }
1618 \DefineFamilyKey{SVG}{extractkeepaspectratio}[true]{%
1619   \FamilySetBool{SVG}{extractkeepaspectratio}{@svg@tempswa}{#1}%
1620   \ifx\FamilyKeyState\FamilyKeyStateProcessed%
1621     \if@svg@tempswa%
1622       \FamilyOptions{SVG}{extractdistort=false}%
1623     \else
1624       \FamilyOptions{SVG}{extractdistort=true}%
1625     \fi%
1626   \else%
1627     \FamilyOptions{SVG}{extractdistort=#1}%
1628   \fi%
1629 }
1630 \newcommand*\svgx@param@scale{\svg@param@scale}
1631 \DefineFamilyKey{SVG}{extractscale}{%
1632   \FamilyKeyStateUnknownValue%
1633   \svg@ifvalueisrelax{#1}{%
1634     \renewcommand*\svgx@param@scale{1}%
1635     \FamilyKeyStateProcessed%
1636   }{%
1637     \ifstr{#1}{inherit}{%
1638       \renewcommand*\svgx@param@scale{\svg@param@scale}%
1639       \FamilyKeyStateProcessed%
1640     }{%
1641       \ifisdimension{#1\p@}{%
1642         \ifdim\dimexpr#1\p@\relax>\z@\relax%
1643           \renewcommand*\svgx@param@scale{#1}%
1644           \FamilyKeyStateProcessed%
1645         \fi%
1646       }{}%
1647     }%
1648   }%
```

```
1649 }
1650 ⟨/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.

```
1651 ⟨*base⟩
1652 \svg@dummy@key{extractpretex}
1653 \svg@dummy@key{extractapptex}
1654 \svg@dummy@key{extractpostex}
1655 ⟨/base⟩
1656 ⟨*extract⟩
1657 \newcommand*\svgx@param@pretex{\svg@param@pretex}
1658 \DefineFamilyKey{SVG}{extractpretex}{%
1659   \svg@ifvalueisrelax{#1}{%
1660     \let\svgx@param@pretex\relax%
1661   }{%
1662     \ifstr{#1}{inherit}{%
1663       \renewcommand*\svgx@param@pretex{\svg@param@pretex}%
1664     }{%
1665       \renewcommand*\svgx@param@pretex{#1}%
1666     }%
1667   }%
1668   \FamilyKeyStateProcessed%
1669 }
1670 \newcommand*\svgx@param@apptex{\svg@param@apptex}
1671 \DefineFamilyKey{SVG}{extractapptex}{%
1672   \svg@ifvalueisrelax{#1}{%
1673     \let\svgx@param@apptex\relax%
1674   }{%
1675     \ifstr{#1}{inherit}{%
1676       \renewcommand*\svgx@param@apptex{\svg@param@apptex}%
1677     }{%
1678       \renewcommand*\svgx@param@apptex{#1}%
1679     }%
1680   }%
1681   \FamilyKeyStateProcessed%
1682 }
1683 \DefineFamilyKey{SVG}{extractpostex}{%
1684   \svg@deprecated@key[svg-extract]{extractpostex=#1}{extractapptex=#1}%
1685 }
1686 ⟨/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.

```
1687 ⟨*base⟩
1688 \svg@dummy@key[true]{clean}
1689 \svg@dummy@key[true]{clear}
1690 ⟨/base⟩
1691 ⟨*extract⟩
1692 \newcommand*\svgx@clean{}
1693 \DefineFamilyKey{SVG}{clean}[true]{%
1694   \FamilySetBool{SVG}{clean}{@svg@tempswa}{#1}%
1695   \ifx\FamilyKeyState\FamilyKeyStateProcessed%
1696     \if@svg@tempswa%
1697       \renewcommand*\svgx@clean{log,aux,dvi,out,ps,eps,pdf,\svgx@latex@ext}%
1698     \else%
1699       \renewcommand*\svgx@clean{}%
1700     \fi%
1701   \else%
1702     \renewcommand*\svgx@clean{#1}%
```

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

```
1708 ⟨∗base⟩
1709 \svg@dummy@key[true]{exclude}
1710 ⟨/base⟩
1711 ⟨∗extract⟩
1712 \DefineFamilyKey{SVG}{exclude}[true]{%
1713    \FamilySetBool{SVG}{exclude}{@svg@tempswa}{#1}%
1714    \ifx\FamilyKeyState\FamilyKeyStateProcessed%
1715        \if@svg@tempswa%
1716            \renewcommand*\svg@input[2][]{%
1717                \if@svgx@run\else%
1718                    \PackageWarning{svg-extract}{%
1719                        The image '##2' was\MessageBreak%
1720                        neither extracted nor included%
1721                    }%
1722                \fi%
1723            }%
1724        \else%
1725            \renewcommand*\svg@input{\svg@@input}%
1726        \fi%
1727    \fi%
1728 }
1729 ⟨/extract⟩
```

C.2. User commands

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

```
1730 ⟨∗extract⟩
1731 \newcommand*\svgx@param@angle{0}
1732 \svg@local@param@def{%
1733    \DefineFamilyKey[.param]{SVG}{extractangle}{%
1734        \FamilyKeyStateUnknownValue%
1735        \svg@ifvalueisrelax{#1}{%
1736            \renewcommand*\svgx@param@angle{0}%
1737            \FamilyKeyStateProcessed%
1738        }{%
1739            \ifstr{#1}{inherit}{%
1740                \renewcommand*\svgx@param@angle{\svg@param@angle}%
1741                \FamilyKeyStateProcessed%
1742            }{%
1743                \ifisdimension{#1\p@}{%
1744                    \renewcommand*\svgx@param@angle{#1}%
1745                    \FamilyKeyStateProcessed%
1746                }{}%
1747            }%
1748        }%
1749    }%
1750 }
1751 ⟨/extract⟩
```

Some dummys for package **svg**.

```
1752 ⟨∗base⟩
1753 \newcommand*\svghidepreamblestart{%
```

```
1754    \PackageWarning{svg}{%
1755      The macro '\string\svghidepreamblestart' is only meant\MessageBreak%
1756      to be used together with package 'svg-extract'.\MessageBreak%
1757      Nevertheless, nothing will happen%
1758    }%
1759 }
1760 \newcommand*\svghidepreambleend{%
1761    \PackageWarning{svg}{%
1762      The macro '\string\svghidepreambleend' is only meant\MessageBreak%
1763      to be used together with package 'svg-extract'.\MessageBreak%
1764      Nevertheless, nothing will happen%
1765    }%
1766 }
1767 ⟨/base⟩
```

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

```
1768 ⟨*extract⟩
1769 \let\svghidepreamblestart\relax
1770 \let\svghidepreambleend\relax
1771 ⟨/extract⟩
```

C.3. Auxiliary macros

\svg@extract
\svgx@stream@in
\svgx@read@line
\svgx@stream@out
\if@svgx@preamble@write

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.

```
1772 ⟨*base⟩
1773 \newcommand*\svg@extract[1]{}
1774 ⟨/base⟩
1775 ⟨*extract⟩
1776 \ifnum\pdf@shellescape=\@ne\relax\else%
1777    \renewcommand*\svg@extract[1]{%
1778      \if@svgx@run%
1779        \begingroup%
1780          \edef\svg@tempa{#1}%
1781          \svg@quotes@remove{\svg@tempa}%
1782          \PackageWarning{svg-extract}{%
1783            You didn't enable 'shell escape' (or 'write18')\MessageBreak%
1784            so it wasn't possible to run the extraction for\MessageBreak%
1785            file '\svg@tempa'\MessageBreak%
1786          }%
1787        \endgroup%
1788      \fi%
1789    }%
1790    \expandafter\endinput%
1791 \fi
```

If --shell-escape is enabled, the command is defined with its intended functionality. Some macros and a input stream as well as a output stream are necessary for this.

```
1792 \newread\svgx@stream@in
1793 \newcommand*\svgx@read@line{}
1794 \newwrite\svgx@stream@out
1795 \newif\if@svgx@preamble@write
1796 \renewcommand*\svg@extract[1]{%
```

If option extract is enabled...

```
1797    \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.

```
1798    \if@svgx@out@sec%
1799      \svg@get@out@sec%
1800    \fi%
1801    \stepcounter{svgx@out@count}%
1802    \begingroup%
1803      \def\svg@tempa##1.##2\@nil{%
1804        \IfArgIsEmpty{##2}{\edef\svgx@preamble{##1.\svgx@latex@ext}}{}%
1805      }%
1806      \expandafter\svg@tempa\svgx@preamble.\@nil%
1807      \IfFileExists{\svg@file@path\svgx@preamble}{%
1808        \@svg@file@foundtrue%
1809      }{%
1810        \svg@get@path[]{\svgx@preamble}{\svg@out@path}%
1811        \def\svg@tempa####1.####2\@nil{%
1812          \edef\svgx@preamble{\svg@file@name.####2}%
1813        }%
1814        \expandafter\svg@tempa\svgx@preamble\@nil%
1815      }%
1816      \edef\svg@tempa{%
1817        \endgroup%
1818        \if@svg@file@found%
1819          \ifx\svg@file@path\@empty%
1820            \def\noexpand\svgx@preamble{./\svgx@preamble}%
1821          \else%
1822            \def\noexpand\svgx@preamble{\svg@file@path\svgx@preamble}%
1823          \fi%
1824        \fi%
1825      }%
1826    \svg@tempa%
1827    \begingroup%
1828      \endlinechar=\m@ne%
1829      \IfFileExists{\svgx@preamble}{%
1830        \PackageInfo{svg-extract}{%
1831          The preamble file '\svgx@preamble'\MessageBreak%
1832          is used for the generation of the auxiliary file\MessageBreak%
1833          '\svgx@out@name.\svgx@latex@ext'%
1834        }%
```

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

```
1835        \catcode`\#=12\relax%
1836        \immediate\openout\svgx@stream@out=\svgx@out@name.\svgx@latex@ext%
1837        \immediate\openin\svgx@stream@in=\svgx@preamble%
1838        \@svg@tempswatrue%
1839        \@svgx@preamble@writetrue%
1840        \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.

```
1841        \@whilesw\if@svg@tempswa\fi{%
1842          \immediate\read\svgx@stream@in to\svgx@read@line%
1843          \ifx\svgx@read@line\@empty%
1844            \ifeof\svgx@stream@in\@svg@tempswafalse\fi%
1845          \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`

```
1846              \svgx@read@preamble@till{\svghidepreamblestart}{}%
1847              \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`.

```
1848              \svgx@read@preamble@till{\svgx@endpreamble}{\@svg@tempswafalse}%
1849              \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.

```
1850              \if@svgx@classfound\else%
1851                \expandafter\svgx@documentclass%
1852                  \svgx@read@line\documentclass\documentclass\@nil%
1853              \fi%
```

Writing out the—maybe manipulated—read in line.

```
1854              \ifx\svgx@read@line\@empty\else%
1855                \immediate\write\svgx@stream@out{%
1856                  \unexpanded\expandafter{\svgx@read@line}%
1857                }%
1858              \fi%
1859            \fi%
1860          \fi%
1861        }%
1862        \immediate\closein\svgx@stream@in%
1863        \immediate\closeout\svgx@stream@out%
1864        \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.

```
1865        \immediate\openin\svgx@stream@in=\svgx@out@name.\svgx@latex@ext%
1866        \def\svg@tempa{}%
1867        \loop\unless\ifeof\svgx@stream@in%
1868          \readline\svgx@stream@in to\svgx@read@line%
1869          \ifx\svgx@read@line\@empty\else%
1870            \edef\svg@tempa{%
1871              \unexpanded\expandafter{\svg@tempa}%
1872              \unexpanded\expandafter{\svgx@read@line}^^J%
1873            }%
1874          \fi%
1875        \repeat%
1876        \immediate\closein\svgx@stream@in%
1877      }{%
```

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

```
1878        \svg@quotes@remove{\svgx@preamble}%
1879        \ifx\svgx@preamble\@empty\else%
1880          \PackageWarning{svg-extract}{%
1881            The preamble file `\svgx@preamble'\MessageBreak%
1882            does not exist%
1883          }%
1884        \fi%
1885        \def\svg@tempa{}%
1886      }%
```

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.

```
1887        \immediate\openout\svgx@stream@out=\svgx@out@name.\svgx@latex@ext%
1888        \immediate\write\svgx@stream@out{%
1889          \@percentchar\@percentchar\space This file was generated by package
1890          `svg-extract'^^J%
1891          \@percentchar\@percentchar\space from source `\jobname'^^J%
1892          \@percentchar\@percentchar\space It's intended to be compiled with
1893          `\svgx@latex@exe\ifx\svgx@latex@opt\@empty\else\space\svgx@latex@opt\fi'
1894        }%
```

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.

```
1895        \immediate\write\svgx@stream@out{%
1896          \string\AtBeginDocument{\@percentchar^^J%
1897            \space\space\string\svgxsetpapersize\@percentchar^^J%
1898            \ifxetex\else\ifpdf\else%
1899              \space\space\string\AtBeginDvi{\string\special{%
1900                  papersize=\string\the\string\paperwidth,%
1901                    \string\the\string\paperheight%
1902              }}\@percentchar^^J%
1903            \fi\fi%
1904          }^^J%
1905          \string\PassOptionsToPackage{hidelinks}{hyperref}%
1906        }%
```

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

```
1907        \if@svgx@classfound\else%
1908          \immediate\write\svgx@stream@out{\string\documentclass{article}}%
1909        \fi%
```

And now the stored preamble.

```
1910        \ifx\svg@tempa\@empty\else%
1911          \immediate\write\svgx@stream@out{\unexpanded\expandafter{\svg@tempa}}%
1912        \fi%
```

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

```
1913        \immediate\write\svgx@stream@out{\string\usepackage{svg-extract}}%
```

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

```
1914        \def\svg@tempa##1{%
1915          \immediate\write\svgx@stream@out{\string\svgsetup{##1}}%
1916        }%
1917        \if@svg@ink@latex\else%
1918          \svg@tempa{inkscapelatex=false}%
1919        \fi%
1920        \ifdim\svgx@param@width>\z@\relax%
1921          \svg@tempa{width=\svgx@param@width}%
1922        \fi%
1923        \ifdim\svgx@param@height>\z@\relax%
1924          \svg@tempa{height=\svgx@param@height}%
1925        \fi%
1926        \if@svgx@param@distort%
1927          \svg@tempa{distort=true}%
1928        \fi%
1929        \ifdim\dimexpr\svgx@param@scale\p@\relax=\p@\relax\else%
```

```
1930        \svg@tempa{scale=\svgx@param@scale}%
1931      \fi%
1932      \def\svg@tempb{\svg@param@pretex}%
1933      \ifx\svgx@param@pretex\svg@tempb\relax%
1934        \let\svgx@param@pretex\svg@param@pretex%
1935      \fi%
1936      \ifx\svgx@param@pretex\relax\else%
1937        \svg@tempa{pretex=\unexpanded\expandafter{\svgx@param@pretex}}%
1938      \fi%
1939      \def\svg@tempb{\svg@param@apptex}%
1940      \ifx\svgx@param@apptex\svg@tempb\relax%
1941        \let\svgx@param@apptex\svg@param@apptex%
1942      \fi%
1943      \ifx\svgx@param@apptex\relax\else%
1944        \svg@tempa{apptex=\unexpanded\expandafter{\svgx@param@apptex}}%
1945      \fi%
```

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

```
1946        \let\svg@tempa\@empty%
1947        \if@svg@ink@latex%
1948          \ifstr{\svg@ink@format}{pdf}{%
1949            \ifnum\value{svg@param@lastpage}>\z@\relax%
1950              \edef\svg@tempa{lastpage=\the\value{svg@param@lastpage}}%
1951            \else%
1952              \ifnum\value{svg@param@lastpage}=\z@\relax%
1953                \def\svg@tempa{lastpage=true}%
1954              \else%
1955                \def\svg@tempa{lastpage=false}%
1956              \fi%
1957            \fi%
1958          }{}%
1959        \fi%
```

The rotation angle, if given.

```
1960        \ifdim\dimexpr\svgx@param@angle\p@\relax=\z@\relax\else%
1961          \edef\svg@tempa{%
1962            angle=\svgx@param@angle\ifx\svg@tempa\@empty\else,\svg@tempa\fi%
1963          }%
1964        \fi%
```

As we are now at 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.

```
1965        \ifx\svg@tempa\@empty%
1966          \def\svg@tempa{\string\svgxsetbox{#1}}%
1967        \else%
1968          \edef\svg@tempa{\noexpand\string\noexpand\svgxsetbox[\svg@tempa]{#1}}%
1969        \fi%
1970        \immediate\write\svgx@stream@out{\svg@tempa}%
```

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

```
1971        \if@svg@ink@latex%
1972          \IfFileExists{xr.sty}{%
1973            \immediate\write\svgx@stream@out{%
1974              \string\usepackage{xr}^^J%
1975              \string\externaldocument{\jobname}^^J%
1976            }%
1977          }{}%
1978        \fi%
1979        \immediate\write\svgx@stream@out{%
1980          \string\begin{document}^^J%
```

```
1981        \string\pagestyle{empty}^^J%
1982        \string\svgxoutputbox\@percentchar^^J%
1983        \string\end{document}%
1984      }%
1985      \immediate\closeout\svgx@stream@out%
1986    \endgroup%
```

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

```
1987    \ifstr{\svgx@format\svgx@cnv@format}{}{%
1988      \PackageWarning{svg-extract}{%
1989        Both keys 'extractformat' and 'convertformat' are\MessageBreak%
1990        empty, so nothing to do so far%
1991      }%
1992    }{%
```

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**.

```
1993      \svg@quotes@remove{\svgx@out@path}%
1994      \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.

```
1995      \edef\svg@tempb{%
1996        \noexpand\svg@shell@mkdir{\svgx@out@path}%
1997      }%
1998      \expandafter\AfterReadingMainAux\expandafter{\svg@tempb}%
```

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

```
1999      \edef\svg@tempb{%
2000        \noexpand\PackageInfo{svg-extract}{%
2001          Running LaTeX (\svgx@latex@exe) for graphic extraction%
2002          \ifx\svgx@latex@opt\@empty\else%
2003            \MessageBreak with added options '\svgx@latex@opt'%
2004          \fi%
2005        }%
2006      }%
2007      \expandafter\AfterReadingMainAux\expandafter{\svg@tempb}%
2008      \edef\svg@tempb{%
2009        \noexpand\ShellEscape{%
2010          \svgx@latex@exe\space\svgx@latex@opt\space%
2011          "\svgx@out@name.\svgx@latex@ext"%
2012        }%
2013      }%
2014      \loop\ifnum\value{svgx@runs}>\z@\relax%
2015        \expandafter\AfterReadingMainAux\expandafter{\svg@tempb}%
2016        \advance\c@svgx@runs\m@ne%
2017      \repeat%
```

All files requested with option `extractformat` are created with internal conversion tools supplied by most LaTeX 2_ε distributions if necessary.

```
2018      \def\svg@tempa##1##2##3{%
2019        \edef\svg@tempb{%
2020          \noexpand\ShellEscape{%
2021            \@nameuse{svgx@##1@exe}\space\@nameuse{svgx@##1@opt}\space%
2022            "\svgx@out@name.##2"%
2023          }%
2024        }%
2025        \AfterReadingMainAux{\PackageInfo{svg-extract}{Running ##1}}%
```

```
2026        \expandafter\AfterReadingMainAux\expandafter{\svg@tempb}%
2027      }%
2028      \@svg@tempswafalse%
2029      \ifxetex\else\ifpdf\else%
2030        \@svg@tempswatrue%
2031      \fi\fi%
2032      \if@svg@tempswa%
2033        \svg@tempa{dvips}{dvi}{ps}%
2034        \svgx@ifinlist{eps}{\svgx@format}{\svg@tempa{pstoeps}{ps}{eps}}{}%
2035        \svgx@ifinlist{pdf}{\svgx@format}{\svg@tempa{pstopdf}{ps}{pdf}}{}%
2036      \else%
2037        \svgx@ifinlist{eps}{\svgx@format}{\svg@tempa{pdftoeps}{pdf}{eps}}{}%
2038        \svgx@ifinlist{ps}{\svgx@format}{\svg@tempa{pdftops}{pdf}{ps}}{}%
2039      \fi%
```

Now the desired conversion tool is invoked if requested.

```
2040        \if@svgx@cnv@run%
```

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

```
2041        \ifx\svgx@cnv@dpi\relax%
2042          \ifx\svgx@cnv@dpi@png\@undefined%
2043            \def\svgx@cnv@dpi@png{300}%
2044          \fi%
2045        \fi%
```

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

```
2046        \expandafter\svgx@cnv@get@informat\expandafter{\svgx@format}%
```

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

```
2047        \@for\svg@tempa:=\svgx@cnv@format\do{%
2048          \ifx\svg@tempa\@empty\else%
2049            \expandafter\svgx@ifinlist\expandafter{\svg@tempa}{\svgx@format}{%
2050              \PackageWarning{svg-extract}{%
2051                File type '\svg@tempa' was specified for option\MessageBreak%
2052                'extractformat' (\svgx@format) as well as for \MessageBreak%
2053                option 'convertformat' (\svgx@cnv@format) so the\MessageBreak%
2054                conversion won't be done%
2055              }%
2056            }{%
2057              \edef\svg@tempb{%
2058                \noexpand\PackageInfo{svg-extract}{%
2059                  Converting '\svgx@out@name.\svgx@cnv@informat'\MessageBreak%
2060                  to '\svgx@out@name.\svg@tempa'%
2061                }%
2062              }%
2063              \expandafter\AfterReadingMainAux\expandafter{\svg@tempb}%
2064              \edef\svg@tempb{%
2065                \noexpand\ShellEscape{%
2066                  \svgx@cnv@cmd{\svgx@out@name}{\svgx@cnv@informat}{\svg@tempa}%
2067                }%
2068              }%
2069              \expandafter\AfterReadingMainAux\expandafter{\svg@tempb}%
2070            }%
2071          \fi%
2072        }%
2073      \fi%
```

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

```
2074        \edef\svg@tempa{\svgx@format\if@svgx@cnv@run,\svgx@cnv@format\fi}%
2075        \@for\svg@tempb:=\svg@tempa\do{%
```

```
2076        \ifx\svg@tempb\@empty\else%
2077          \edef\svg@tempb{%
2078            \noexpand\svgx@move{\svgx@out@name}{\svg@tempb}{\svgx@out@path}%
2079          }%
2080          \expandafter\AfterReadingMainAux\expandafter{\svg@tempb}%
2081        \fi%
2082      }%
```

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

```
2083      \@for\svg@tempa:=\svgx@clean\do{%
2084        \expandafter\svgx@ifinlist\expandafter{\svg@tempa}{\svg@tempb}{}{%
2085          \edef\svg@tempb{%
2086            \noexpand\IfFileExists{"\svgx@out@name".\svg@tempa}{%
2087              \noexpand\svg@shell@rm{\svgx@out@name.\svg@tempa}%
2088            }{}%
2089          }%
2090          \expandafter\AtEndDocument\expandafter{%
2091            \expandafter\AfterReadingMainAux\expandafter{\svg@tempb}%
2092          }%
2093        }%
2094      }%
2095    }%
2096  \fi%
2097 }
2098 ⟨/extract⟩
```

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

```
2099 \newcommand*\svgx@out@sec{unknown}
2100 \newcommand*\svgx@get@out@sec{%
2101   \begingroup%
2102     \def\svg@tempa{}%
2103     \@for\svg@tempb:={%
2104       part,chapter,section,subsection,subsubsection,paragraph,subparagraph%
2105     }\do{%
2106       \ifx\svg@tempb\@empty\else%
2107         \scr@ifundefinedorrelax{the\svg@tempb}{}{%
2108           \ifnum\value{\svg@tempb}>\z@\relax%
2109             \edef\svg@tempa{\svg@tempb}%
2110           \fi%
2111         }%
2112       \fi%
2113     }%
2114     \edef\svg@tempb{%
2115       \endgroup%
2116       \ifx\svg@tempa\@empty\else%
2117         \def\noexpand\svgx@out@sec{\csname the\svg@tempa\endcsname}%
2118       \fi%
2119     }%
2120   \svg@tempb%
2121 }
```

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

```
2122 \newif\if@svgx@classfound
2123 \newcommand*\svgx@documentclass{}
2124 \def\svgx@documentclass#1\documentclass#2\documentclass#3\@nil{%
2125   \IfArgIsEmpty{#2}{}{\@svgx@classfoundtrue}%
2126 }
```

`\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.

```
2127 \newcommand*\svgx@read@preamble@till[2]{%
2128   \svgx@read@preamble@skip#1\@nil{till}{#2}%
2129 }
2130 \newcommand*\svgx@read@preamble@from[2]{%
2131   \svgx@read@preamble@skip#1\@nil{from}{#2}%
2132 }
```

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

```
2133 \newcommand*\svgx@read@preamble@skip{}
2134 \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.

```
2135   \def\svg@tempa##1{%
2136     \def\svg@tempa####1##1####2##1####3\@nil{%
2137       \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`.

```
2138         \ifstr{#2}{till}{%
2139           \IfArgIsEmpty{####1}{}{%
2140             \immediate\write\svgx@stream@out{####1}%
2141           }%
2142           \@svgx@preamble@writefalse%
2143         }{%
```

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

```
2144         \ifstr{#2}{from}{%
2145           \IfArgIsEmpty{####2}{%
2146             \def\svgx@read@line{}%
2147           }{%
2148             \def\svgx@read@line{####2}%
2149           }%
2150           \@svgx@preamble@writetrue%
2151         }{}%
2152       }%
```

Additonal stuff which should be done.

```
2153         #3%
2154       }%
2155     }%
2156   }%
```

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

```
2157   \edef\svg@tempb{\expandafter\detokenize\expandafter{#1}}%
2158   \expandafter\svg@tempa\expandafter{\svg@tempb}%
```

Calling the created macro.

```
2159   \edef\svg@tempb{%
2160     \expandafter\detokenize\expandafter{\svgx@read@line}\svg@tempb\svg@tempb%
2161   }%
2162   \expandafter\svg@tempa\svg@tempb\@nil%
2163 }
```

61

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

```
2164 \newcommand*\svgx@cnv@informat{}
2165 \newcommand*\svgx@cnv@get@informat[1]{%
2166   \begingroup%
2167     \def\svg@tempa##1,##2\@nil{%
2168       \def\svg@tempa{##1}%
2169     }%
2170     \svg@tempa#1,\@nil%
2171     \edef\svg@tempa{%
2172       \endgroup%
2173       \def\noexpand\svgx@cnv@informat{\svg@tempa}%
2174     }%
2175   \svg@tempa%
```

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

```
2176   \ifx\svgx@cnv@informat\@empty%
2177     \renewcommand*\svgx@cnv@informat{pdf}%
2178     \ifxetex\else\ifpdf\else%
2179       \renewcommand*\svgx@cnv@informat{ps}%
2180     \fi\fi%
2181   \fi%
2182 }
```

Depending on option `convert`, one of these two macros is actually used by \svgx@cnv@cmd. For invoking the conversion process, the required platform-dependent executable is set, if nothing was set by a package option.

```
2183 \ifx\svgx@magick@exe\@empty
2184   \ifwindows
2185     \renewcommand*\svgx@magick@exe{magick}
2186   \else
2187     \renewcommand*\svgx@magick@exe{convert}
2188   \fi
2189 \fi
2190 \newcommand*\svgx@magick@cmd[3]{%
2191   \svgx@magick@exe\space%
2192   \svgx@useformatkey{svgx@cnv@dpi}{#3}{-density }%
2193   \svgx@useformatkey{svgx@magick@set}{#3}{}%
2194   "#1.#2"\space%
2195   \svgx@useformatkey{svgx@magick@opr}{#3}{}%
2196   "#1.#3"%
2197 }
```

```
2198 \ifx\svgx@gs@exe\@empty
2199   \ifwindows
2200     \renewcommand*\svgx@gs@exe{gswin64c}
2201   \else
2202     \renewcommand*\svgx@gs@exe{gs}
2203   \fi
2204 \fi
2205 \newcommand*\svgx@gs@cmd[3]{%
2206   \svgx@gs@exe\space-dSAFER -dBATCH -dNOPAUSE\space%
2207   \svgx@useformatkey{svgx@gs@device}{#3}{-sDEVICE=}%
2208   \svgx@useformatkey{svgx@cnv@dpi}{#3}{-r}%
2209   \svgx@useformatkey{svgx@gs@opt}{#3}{}%
2210   -sOutputFile="#1.#3"\space"#1.#2"%
2211 }
```

If the file doesn't exist

```
2212 \newcommand*\svgx@move[3]{%
2213   \begingroup%
```

```
2214      \IfFileExists{"#1".#2}{%
2215        \svg@shell@move{#1.#2}{#3#1.#2}%
2216      }{%
2217        \edef\svg@tempa{#2}%
2218        \@svg@tempswafalse%
2219        \expandafter\svgx@ifinlist\expandafter{\svg@tempa}{\svgx@cnv@format}{%
2220          \@svg@tempswatrue%
2221          \def\svg@tempb{conversion}%
2222        }{%
2223          \expandafter\svgx@ifinlist\expandafter{\svg@tempa}{pdf,ps,eps}{%
2224            \@svg@tempswatrue%
2225            \def\svg@tempb{extraction}%
2226          }{}%
2227        }%
2228        \if@svg@tempswa%
2229          \edef\svg@tempb{%
2230            The graphic file \svg@tempb\space failed\MessageBreak%
2231            for '#1.#2'\MessageBreak%
2232            Troubleshooting: Please check the log file how\MessageBreak%
2233            the invocation of the extraction took place and\MessageBreak%
2234            try to execute it yourself in the terminal%
2235          }%
2236        \else%
2237          \def\svg@tempb{%
2238            The extraction to format '#2' failed\MessageBreak%
2239            for '#1.#2'\MessageBreak%
2240            Only file types 'pdf,ps,eps' are supported for\MessageBreak%
2241            key 'exportformat'%
2242          }%
2243        \fi%
2244        \PackageWarning{svg-extract}{\svg@tempb}%
2245      }%
2246    \endgroup%
2247 }
```

\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.

```
2248 \newcommand*\svgx@ifinlist[2]{%
2249   \begingroup%
2250     \def\svg@tempa##1,#1,##2\@nil{%
2251       \IfArgIsEmpty{##2}{%
2252         \aftergroup\@secondoftwo%
2253       }{%
2254         \aftergroup\@firstoftwo%
2255       }%
2256     }%
2257     \expandafter\svg@tempa\expandafter,#2,#1,\@nil%
2258   \endgroup%
2259 }
```

\svgx@onlywindows   Do only some stuff, if Windows was detected.

```
2260 \newcommand*\svgx@onlywindows[1]{}
2261 \AfterPackage*{ifplatform}{\renewcommand*\svgx@onlywindows[1]{\ifwindows#1\fi}}
```

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

```
2262 \newcommand*\svgx@ifkeyandval[3]{%
2263   \def\svg@tempa##1=##2=##3\@nil{\ifstr{##3}{=}{#2}{#3}}%
2264   \svg@tempa#1==\@nil%
2265 }
```

\svgx@cnv@get@dpi 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.

```
2266 \newcommand*\svgx@cnv@get@dpi[1]{%
2267   \begingroup%
2268     \def\svg@tempa##1dpi##2x##3dpi##4\@nil{%
2269       \edef\svg@tempa{##1}%
```

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

```
2270       \@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.

```
2271       \ifnumber{##1}{%
2272         \IfArgIsEmpty{##3}{\@svg@tempswatrue}{%
2273           \ifnumber{##3}{\edef\svg@tempa{##1x##3}}{}%
2274         }%
2275       }{}%
2276       \if@svg@tempswa\else%
2277         \expandafter\svg@tempb\svg@tempa xx\@nil%
2278       \fi%
2279     }%
```

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

```
2280     \def\svg@tempb##1x##2x##3\@nil{%
2281       \ifstr{##3}{x}{%
2282         \@svg@tempswatrue%
2283         \IfArgIsEmpty{##1}{\@svg@tempswafalse}{%
2284           \ifnumber{##1}{}{\@svg@tempswafalse}%
2285         }%
2286         \IfArgIsEmpty{##2}{\@svg@tempswafalse}{%
2287           \ifnumber{##2}{}{\@svg@tempswafalse}%
2288         }%
2289         \if@svg@tempswa%
2290           \edef\svg@tempa{##1x##2}%
2291         \fi%
2292       }{}%
2293     }%
2294     \IfArgIsEmpty{#1}{%
2295       \let\svg@tempa\@empty%
2296     }{%
2297       \lowercase{\svg@tempa#1dpi#1xdpi\@nil}%
2298       \if@svg@tempswa\else%
2299         \let\svg@tempa\relax%
2300       \fi%
2301     }%
2302     \edef\svg@tempb{%
2303       \endgroup%
2304       \ifx\svg@tempa\relax%
2305         \let\noexpand\svg@tempa\noexpand\relax%
2306       \else%
2307         \def\noexpand\svg@tempa{\svg@tempa}%
2308       \fi%
2309     }%
2310   \svg@tempb%
2311 }
```

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.

```
2312 \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.

```
2313   \svgx@ifkeyandval{#1}{%
2314     \svg@ifvalueisrelax{##2}{%
2315       \expandafter\let\csname #2@##1\endcsname\relax%
2316     }{%
2317       \@namedef{#2@##1}{##2}%
2318     }%
```

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

```
2319   }{%
2320     \svg@ifvalueisrelax{##1}{%
2321       \expandafter\let\csname #2\endcsname\relax%
2322     }{%
2323       \@namedef{#2}{##1}%
2324     }%
2325   }%
2326 }
```

The command \svgx@useformatkey checks, if a format specific key was definded 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.

```
2327 \newcommand*\svgx@useformatkey[3]{%
2328   \scr@ifundefinedorrelax{#1@#2}{%
2329     \scr@ifundefinedorrelax{#1}{}{%
2330       \expandafter\ifx\csname #1\endcsname\@empty\else%
2331         #3\@nameuse{#1}\space%
2332       \fi%
2333     }%
2334     \scr@ifundefinedorrelax{#1@#2+}{}{%
2335       \expandafter\ifx\csname #1@#2+\endcsname\@empty\else%
2336         #3\@nameuse{#1@#2+}\space%
2337       \fi%
2338     }%
2339   }{%
```

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

```
2340     \expandafter\ifx\csname #1@#2\endcsname\@empty\else%
2341       #3\@nameuse{#1@#2}\space%
2342     \fi%
2343   }%
2344 }
```

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.

Within the preamble of the auxiliary LATEX file, the desired grahic 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 behavoiur for **svg-extract** in standalone mode. for example, TUD-Script-classes are using this switch.

```
2345 \newif\if@svgx@standalone
2346 \newcommand*\svgxsetbox[2][]{%
2347   \@svgx@standalonetrue%
2348   \svgx@setbox{#1}{#2}%
2349   \scr@ifundefinedorrelax{AtEndPreamble}{%
2350     \let\svg@tempa\@firstofone%
2351   }{%
2352     \def\svg@tempa{\AtEndPreamble}%
2353   }%
2354   \svg@tempa{\AtBeginDocument{\svgx@setbox{#1}{#2}}}%
2355 }
2356 \newcommand*\svgx@setbox[2]{%
2357   \sbox\svg@box{\svg@@input[{#1},draft=false]{#2}}%
2358   \svgxsetpapersize%
2359 }
```

`\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`.

```
2360 \newcommand*\svgxsetpapersize{%
2361   \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`.

```
2362   \scr@ifundefinedorrelax{stockwidth}{}{%
2363     \setlength\stockwidth{\paperwidth}%
2364   }%
2365   \scr@ifundefinedorrelax{mediawidth}{}{%
2366     \setlength\mediawidth{\paperwidth}%
2367   }%
2368   \setlength\textwidth{\paperwidth}%
2369   \setlength\paperheight{\the\dimexpr\ht\svg@box+\dp\svg@box\relax}%
2370   \scr@ifundefinedorrelax{stockheight}{}{%
2371     \setlength\stockheight{\paperheight}%
2372   }%
2373   \scr@ifundefinedorrelax{mediaheight}{}{%
2374     \setlength\mediaheight{\paperheight}%
2375   }%
2376   \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.

```
2377   \hoffset=-1in%
2378   \oddsidemargin=\z@%
2379   \evensidemargin=\z@%
2380   \voffset=-1in%
2381   \topmargin=\z@%
2382   \headheight=\z@%
2383   \headsep=\z@%
2384   \topskip=\z@%
2385   \footskip=\z@%
2386   \marginparsep=\z@%
2387   \marginparwidth=\z@%
2388   \marginparpush=\z@%
2389 }
2390 \@onlypreamble\svgxsetpapersize
```

`\svgxoutputbox` With `\svgxoutputbox` the created box is displayed.
`\if@svgx@beamer`

```
2391 \newif\if@svgx@beamer
2392 \@ifclassloaded{beamer}{\@svgx@beamertrue}{}%
2393 \newcommand*\svgxoutputbox{%
2394   \begingroup%
2395     \setlength\parindent{\z@}%
2396     \setlength\parskip{\z@}%
2397     \setlength\parfillskip{\z@}%
2398     \if@svgx@beamer%
2399       \setbeamertemplate{navigation symbols}{}%
2400       \begin{frame}[plain]%
2401       \usebox\svg@box%
2402       \end{frame}%
2403     \else%
2404       \usebox\svg@box%
2405     \fi%
2406     \endgraf%
2407   \endgroup%
2408 }
```

D. Processing Options

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

```
2409 ⟨*base⟩
2410 \FamilyExecuteOptions{SVG}{%
2411   inkscape=true,inkscapepath=basesubdir,
2412   inkscapelatex=true,inkscapearea=drawing,distort=false,%
2413   usexcolor=true,usetransparent=true%
2414 }
2415 ⟨/base⟩
2416 ⟨*extract⟩
2417 \FamilyExecuteOptions{SVG}{%
2418   extract=true,extractpath=basesubdir,%
2419   extractruns=2,extractname=namenumbered,extractdistort=false,%
2420   convert=magick,convert=false,%
2421   gsdevice={png=png16m},gsdevice={jpeg=jpeg},gsdevice={jpg=jpeg},%
2422   gsdevice={tif=tiff48nc},gsdevice={tiff=tiff48nc},%
2423   gsdevice={eps=eps2write},gsdevice={ps=ps2write}%
2424 }
2425 ⟨/extract⟩
2426 \FamilyProcessOptions{SVG}
```

E. Macros for file access

Finally, platform dependend macros for creating directories as well as moving and deleting files are provided, if --shell-escape is enabled. Only then package **ifplatform** is only used in order to do not raise a warning.

```
2427 \ifnum\pdf@shellescape=\@ne\relax\else%
2428   \expandafter\endinput%
2429 \fi
2430 \RequirePackage{ifplatform}[2010/10/22]
```

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

The platform dependent commands for file access.

```
2431 \ifwindows
2432   \newcommand*\svg@shell@@mkdir[1]{if not exist "#1" mkdir "#1"}
2433   \newcommand*\svg@shell@@mv{move}
2434   \newcommand*\svg@shell@@rm{del}
2435 \else
2436   \newcommand*\svg@shell@@mkdir[1]{mkdir -p "#1"}
```

67

```
2437  \newcommand*\svg@shell@@mv{mv}
2438  \newcommand*\svg@shell@@rm{rm}
2439 \fi
```

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

```
2440 \newcommand*\svg@shell@mkdir[1]{%
2441   \begingroup%
2442     \svg@quotes@remove[{#1}]{\svg@tempa}%
2443     \@svg@tempswatrue%
2444     \ifstr{\svg@tempa}{}{\@svg@tempswafalse}{%
2445     \ifstr{\svg@tempa}{./}{\@svg@tempswafalse}{%
2446     }}%
2447     \if@svg@tempswa%
2448       \ShellEscape{\svg@shell@@mkdir{\svg@tempa}}%
2449     \fi%
2450   \endgroup%
2451 }
```

Commands for moving and deleting files.

```
2452 \newcommand*\svg@shell@move[2]{%
2453   \ShellEscape{\svg@shell@@mv\space"#1"\space"#2"}%
2454 }
2455 \newcommand*\svg@shell@rm[1]{%
2456   \ShellEscape{\svg@shell@@rm\space"#1"}%
2457 }
```

At the very end, the catcodes are restored.

```
2458 \svg@catcodecodes@restore
```

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 1687
    convert (opt.): changed/extended . . 1314

v2.00a

Implementation

v2.00b

Implementation

v2.01

General

Implementation

v2.02

General

Implementation

v2.02a

General
  fix bug for package **polyglossia** which
    fakes **babel** poorly *2*
Implementation
  `\svg@deactivate@dq`: bug fix for
    polyglossia . . . . [351](#)

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** . [1033](#)