The Dvipdfmx user manual

The Dvipdfmx project team

February 18, 2026

`https://ctan.org/pkg/dvipdfmx`

Contents

Chapter 1

Getting Started

1.1 Introduction

The `dvipdfmx` (formerly `dvipdfm`-cjk) project provides an extended version of the `dvipdfm`, a DVI to PDF translator developed by Mark A. Wicks.

The primary goal of this project is to support multi-byte character encodings and large character sets such as those for East Asian languages. This project started as a combined work of the `dvipdfm`-jpn project by Shunsaku Hirata and its modified one, `dvipdfm`-kor, by Jin-Hwan Cho.

Extensions to `dvipdfm` include,

- Support for OpenType and TrueType fonts, including partial support for OpenType Layout features for glyph variants and for vertical writing.
- Support for CJK-LaTeX and HLaTeX with Subfont Definition Files.
- Support for various legacy multi-byte encodings via PostScript CMap Resources.
- Unicode related features: Unicode as an input encoding and auto-creation of ToUnicode CMaps.
- Support for pTeX (a Japanese localized variant of TeX) including vertical writing extension.
- Some extended DVI specials.
- Reduction of output files size with on-the-fly Type1 to CFF (Type1C) conversion and PDF object stream.
- Advanced raster image support including alpha channels, embedded ICC profiles, 16-bit bit-depth colors, and so on.
- Basic PDF password security support. (only for output)

Some important features are still missing:

- Linearization.
- Color Management.
- Resampling of images.

- Selection of compression filters.
- Variable font and OpenType 1.8.
- and plenty more...

`dvipdfmx` is now maintained as part of TEX Live. Latest source code can be found at the TEX Live SVN repository. For an instruction on accessing the development sources for TEX Live, see,

`https://tug.org/texlive/svn/`

Please send bug reports, questions, or suggestions to the public mailing list, `tex-k@tug.org`. For more information, see `https://ctan.org/pkg/dvipdmx`.

1.1.1 `xdvipdfmx`

`xdvipdfmx` is an extended version of `dvipdfmx`, and is now incorporated into `dvipdfmx`.

The `xdvipdfmx` extensions provides support for the Extended DVI (.xdv) format generated by XƎTEX which includes support for platform-native fonts and the XƎTEX graphics primitives, as well as Unicode text and OpenType font.

XƎTEX originally used a Mac-specific program called `xdv2pdf` as a backend program instead of `xdvipdfmx`. The `xdv2pdf` program supported some special effects that are not yet available through `xdvipdfmx`: Quartz graphics-based shadow support, AAT "variation" fonts such as Skia, transparency as a font attribute, and so on.

1.1.2 Legal Notice

1.2 Installation and Usage

`dvipdfmx` is included in the TEX distributions, so usually you don't have to install it separately.

If you do wish to install it separately, typical usage and installation steps are not different from the original `dvipdfm`. Please refer documents from `dvipdfm` distribution for detailed instruction on how to install and how to use `dvipdfm`. The `dvipdfm` manual is available from its CTAN site:

`https://ctan.org/tex-archive/dviware/dvipdfm`

The minimal requirements for building `dvipdfmx` is the *kpathsea* library. the *zlib* library for compression and the *libpng* library for PNG inclusion are highly recommended. Optionally, the *libpaper* library might be used to handle paper size.

This document mainly focuses on the additions and modifications to `dvipdfm`. Please refer the "Dvipdfm User's Manual" available from the CTAN site mentioned above for basic usage.

Some additional command line options recognized by `dvipdfmx` are listed in Table 1.1. In addition to this, the `-V` option for specifying the output PDF version now accepts the version specification of a form `2.0`. Please try

Option	Description
`-C` *number*	Specify miscellaneous option flags. See, section of “Incompatible Changes” for details.
`-S`	Enable PDF encryption.
`-K` *number*	Set encryption key length. The default value is 40.
`-P` *number*	Set permission flags for PDF encryption. The *number* is a 32-bit unsigned integer representing permission flags. See, section of “Encryption Support”. The default value is `0x003C`.
`-I` *number*	Life of image cache in hours, relevant only when an image not directly supported by `dvipdfmx` is used thus an external program is invoked to convert it to a PDF format intermediate file. This option basically specifies how long such intermediate files are preserved and reused. (to avoid an external program is invoked again and again whenever `dvipdfmx` tries to include images) By specifying a value of `0`, `dvipdfmx` erases existing cached images, and the value `-1` tells `dvipdfmx` to erase all cached images and not to leave newly generated one. And `-2` indicates “ignore image cache“. The default value is `-2`.
`-M`	Process METAPOST generated PostScript file.
`-E`	Always try to embed fonts *regardless of licensing.*
`-O` *number*	Set maximum depth of open bookmark item.

Table 1.1: Additional command line options recognized by `dvipdfmx`.

```
dvipdfmx --help
```

for a complete list of command line options and their explanations.

1.3 Quick Guide

As the primary goal of `dvipdfmx` is to support multi-byte character encodings and large character sets, its primary users are expected to be users of LaTeX packages for typesetting CJK languages such as HLaTeX and CJK-LaTeX, and users of extended TeX variants which are capable of handling those languages, like XeTeX, pTeX, and upTeX. This section provides a “Quick Guide” for those users.

1.3.1 XeTeX

XeTeX users normally do not invoke the `dvipdfmx` command directly. To control the behavior of `dvipdfmx`, please consider using the `dvipdfmx:config` special explained in the section of “Specials”. Some part of this document is irrelevant for XeTeX users.

1.3.2 pTEX

pTEX users are at least required to install several auxiliary files mentioned in the section of "Auxiliary Files" and to setup font-mappings. Just install the *adobemappings* and *glyphlist* for auxiliary files. (As TEX Live basic installation requires them, they are probably already installed for TEX Live users.)

For TEX Live users, setting up fontmaps can be easily done with the help of the *ptex-fontmaps* package and the *updmap* program. To use with the IPAex fonts (contained in the *ipaex* package), for example, run,

```
kanji-config-updmap --sys ipaex
```

where the '`--sys`' option indicates the system-wide configuration. After successful invocation of the above command, the IPAex fonts will be used by default. The current setting can be checked via,

```
kanji-config-updmap --sys status
```

For more information on the updmap program, try,

```
kanji-config-updmap --help
```

or refer the documentation of the updmap program.

Alternatively, just for a quick test of installation, try the following:

```
\documentclass{article}
\begin{document}
\special{pdf:mapline rml H KozMinProVI-Regular}
...Some Japanese text goes here...
\end{document}
```

In this example, PDF viewer which can handle substitute font is required since `dvipdfmx` does not embed fonts.

For using Japanese text in PDF document information and annotations, put the following `special` command,

```
\AtBeginDocument{\special{pdf:tounicode 90ms-RKSJ-UCS2}}
```

in the preamble. The above `special` command instructs `dvipdfmx` to convert text encoded in Shift-JIS to Unicode. For EUC-JP, replace 90ms-RKJK-UCS2 with EUC-UCS2.

1.3.3 upTEX

upTEX users are basically the same as pTEX users but there are two choices for setting fontmaps. Setup fontmaps as mentioned above for pTEX, or use keyword `unicode` in the encoding field of the fontmap file.

The former might be easier as the auto-creation of fontmap files can be done with the updmap program and the *ptex-fontmaps* package. But in this method there are some difficulties when using fonts which employ character collections (glyph repertoire) other than Adobe-Japan1 in the case of PostScript flavored OpenType fonts. In the later case, the *adobemappings* package is not required and newer PostScript flavored OpenType fonts which do not employ Adobe-Japan1 can be easily used too.

Using `unicode` is more simpler and intuitive thus it is recommended to use this method.[1] A typical example fontmap entries for using Adobe's SouceHan series might look like:

```
urml    unicode  SourceHanSerifJP-Light.otf
urmlv   unicode  SourceHanSerifJP-Light.otf  -w 1
ugbm    unicode  SourceHanSansJP-Medium.otf
ugbmv   unicode  SourceHanSansJP-Medium.otf  -w 1
```

As in pTEX, the following `special` instruction might be necessary for PDF document information and annotations to be shown correctly:

```
\AtBeginDocument{\special{pdf:tounicode UTF8-UCS2}}
```

Here, input encoding is assumed to be UTF-8.

1.3.4 CJK-LATEX

CJK-LATEX users are required to have *Subfont Definition Files* to be installed. They are available as part of the *ttfutils* package.

To use TrueType Arphic fonts provided by the *arphic-ttf* package:

```
\documentclass{article}
\usepackage{CJKutf8}
...Other packages loaded here...
\AtBeginDocument{%
  \special{pdf:tounicode UTF8-UCS2}%
  \special{pdf:mapline bsmiu@Unicode@ unicode bsmi00lp.ttf}%
 }
\begin{document}
\begin{CJK}{UTF8}{bsmi}
...some Chinese text goes here...
\end{CJK}
\end{document}
```

[1]For TEX Live 2017. Earlier versions have buggy support.

「こんにちは」 「こんにちは」

Figure 1.1: An example of horizontal and vertical text; left and right corner brackets are replaced with their vertical counterparts.

Here, `pdf:mapline` special is used to setup a font-mapping.

1.4 Overview of Extensions

This section gives a quick overview of `dvipdfmx`'s extended capabilities.

1.4.1 CJK Support

There are many extensions made for supporting CJK languages. Features described here are mainly for CJK languages. However, those features are implemented in a more generic way and hence they can be also beneficial to users who are not involved in CJK languages.

Legacy Multi-byte Encodings

`dvipdfmx` has an extensible support for multi-byte encodings by means of PostScript CMap Resources. Just like various 8-bit encodings can be supported via `enc` file, various multi-byte encodings (including custom one) can be supported by preparing CMap files. See, Adobe's technical notes[2] for details on PostScript CMap Resources.

Vertical Writing

`dvipdfmx` supports the vertical writing extension used by pTEX and upTEX. A DVI instruction to set the writing mode is supported. The OpenType Layout GSUB Feature is supported for selecting vertical version of glyphs.

1.4.2 Unicode Support

Unicode support here consists of two parts: Supporting Unicode as an input encoding and making output PDF files "Unicode aware" ("ToUnicode CMap Support").

Unicode as Input Encoding

`dvipdfmx` recognizes an additional keyword `unicode` in the encoding entry of fontmap file, which declares that character code used in input DVI files for fonts with this keyword specified should be regarded as Unicode values. Unicode support is basically limited to the Basic Multilingual Plane (BMP) since there are no support for code ranges that requires more than two bytes in TFM and extended TFM formats.

ToUnicode CMap Support

In PDF, it is often the case that text is not encoded in Unicode. However, modern applications usually want them represented in Unicode to make it usable as text information. The ToUnicode CMap is a bridge between PDF text string encodings and Unicode encodings, and makes it possible to extract text in PDF files as Unicode encoded strings. It is important to make resulting PDF search-able and copy-and-past-able. Dvipdfmx supports auto-creation of ToUnicode CMaps.

It will not work properly for multiply encoded glyphs due to fundamental limitations of Unicode conversion mechanism with ToUnicode CMaps.

1.4.3 Other Extensions

`dvipdfmx` can generate encrypted PDF documents to protect its contents from unauthorized access. It is limited to password-based authentication, and public-key based authentication is not supported. The 256-bit AES encryption is also supported for PDF version 1.7 and 2.0 setting although it may not be supported by PDF viewers.

There are various other improvements over `dvipdfm`. The most notable one is more improved PDF input and output support: The cross-reference stream and object stream introduced in PDF-1.5 are also supported.

Chapter 2

Auxiliary Files

This chapter describes various auxiliary files required for supporting legacy encodings and legacy font format such as PostScript Type1 font. XƎTEX users may skip this chapter.

`dvipdfmx` can handle various input encodings, from 7-bit encodings to variable-width multi-byte encodings. It also has some sort of Unicode support. Several auxiliary files which are not common to TEX users are needed to utilize those features. This chapter shortly describes about those auxiliary files.

2.1 PostScript CMap Resources

PostScript CMap Resources[1] are required for supporting legacy encodings such as Shift-JIS, EUC-JP, Big5, and other East Asian encodings. `dvipdfmx` internally identifies glyphs with identifiers (CIDs[2]) represented as an integer ranging from 0 to 65535 in the CID-based glyph access. PostScript CMap Resources describes the mapping between sequences of input character codes and CIDs. `dvipdfmx` has an extensible support for multi-byte encodings via PostScript CMap Resources.

CMap files for standard East Asian encodings, for use with Adobe's character collections, are included in the *adobemapping* package. The latest version of those CMap files maintained by Adobe can be found at Adobe's GitHub Project page:

`https://github.com/adobe-type-tools/cmap-resources`

Those files are mandatory for supporting pTEX. upTEX users may also want to install them but they are not required.

2.1.1 Subfont Definition Files

CJK fonts usually contain several thousands of glyphs. For using such fonts with (original) TEX, which can only handle 8-bit encodings, it is necessary to split a font into several *subfonts*. The Subfont Definition File (SFD) specify the way how those fonts are split into subfonts. `dvipdfmx` uses SFD files to convert a set of subfonts back to a single font.

SFD files are not required for use with TEX variants which can handle multi-byte character encodings and large character sets such as pTEX, upTEX,XƎTEX, and Omega. HLATEX and CJK-LATEX users are required to have those files to be installed. SFD files are available as a part of the *ttfutils* package for TEX Live users.

[1]See, "Adobe CMap and CIDFont Files Specification"

[2]PostScript terminology "Character IDentifier".

2.1.2 The Adobe Glyph List and ToUnicode Mappings

The Adobe Glyph List[3] (AGL) describes correspondence between PostScript glyph names (e.g., `AE`, `Aacute`,...) and Unicode character sequences representing them. Some features described in the section “Unicode Support” requires AGL file.

`dvipdfmx` looks for the file `glyphlist.txt` when conversion from PostScript glyph names to Unicode sequences is necessary. This conversion is done in various situations; when creating ToUnicode CMaps for 8-bit encoding fonts, finding glyph descriptions from TrueType and OpenType fonts when the font itself does not provide a mapping from PostScript glyph names to glyph indices (version 2.0 “post” table), and when the encoding `unicode` is specified for Type1 font.

The AGL file is included in the *glyphlist* package. The latest version can be found at Adobe’s GitHub site:

```
https://github.com/adobe-type-tools/agl-aglfn
```

ToUnicode Mappings are similar to AGL but they describe correspondence between CID numbers (instead of glyph names) and Unicode values. The content of those files are the same as CMap Resources. They are required when using TrueType fonts emulated as a CID-keyed font. They should be found in the same directory as ordinary CMap files.

ToUnicode Mapping files are included in the *adobemapping* package. Those files are not required for XƎTEX users.

[3]See, “Adobe Glyph List Specification”

Chapter 3

Graphics

3.1 Image Inclusion

The basics of incorporating images into output PDF is the same as in `dvipdfm`. To do this, LaTeX users can simply use the *graphicx* package. (possibly with the driver option `dvipdfmx`) This section is *not* for providing a how-to guide to include images but just for supported graphics and image formats along with supported features.

Graphics support was mostly rewritten in `dvipdfmx`. Support for BMP and JPEG2000 was added. An effort to preserve more information originally found in included images, e.g., embedded ICC Profiles and XMP Metadata, was made.

However, `dvipdfmx` does not support various features common to graphics manipulation programs such as resampling, color conversion, and selection of compression filters. Thus, it is recommended to use other programs specialized in image manipulation for preparation of images.

3.1.1 Supported Graphics File Formats

Supported formats are, PNG, JPEG, JPEG2000, BMP, PDF, and METAPOST generated EPS. All other format images, such as SVG and PostScript, must be converted to PDF before inclusion. The '`-D`' option, as in `dvipdfm`, can be used for filtering images.

Notes on PNG Support

PNG is supported as in `dvipdfm` with many improvements.

PNG support includes most of important features of PNG format such as color palette, transparency, 16-bit bit-depth color, embedded ICC Profiles, calibrated color, and embedded XMP Metadata.

In including PNG images, `dvipdfmx` first decompresses image data and then compresses (if requested) it again. For better compression ratio, a preprocessing filter (Predictor filter) might be applied before compression. `dvipdfmx` supports the TIFF Predictor 2 and the PNG optimum filter. However, there is yet no way to specify which predictor function is to be used and currently PNG optimum filter is always employed.

Predictor filter is a preprocessing filter to image data for improving compression. Using a predictor filter usually gives better compression but in many cases compression speed becomes significantly slower. Try '`-C 0x20`' command line option to disable predictor filters and to check if slowness is due to filtering.

Feature	PDF Version Required
16-bit Color Depth	Version 1.5
Transparency	Full support for alpha channel requires PDF version 1.4. Color key masking (a specific color is treated as fully transparent) requires 1.3.
XMP Metadata	Version 1.4
ICC Profile	Version 1.3

Table 3.1: PNG features and corresponding PDF versions required.

For the PNG optimum filter, a heuristic approach, “minimum sum of absolute differences”, is employed for finding the most optimal filter to be used. See, discussion in the PNG Specification ”Filter selection”:

`https://www.w3.org/TR/2003/REC-PNG-20031110/#12Filter-selection`

As built-in support for the sRGB color space is absent in PDF, the sRGB color can only be represented precisely by means of the sRGB ICC Profile. However, for sRGB color PNG images, `dvipdfmx` uses an approximate calibrated RGB color space instead. For approximating the sRGB color, the gamma and CIE 1931 chromaticity values mentioned in the PNG Specification is used. See, the following page for more information:

`https://www.w3.org/TR/2003/REC-PNG-20031110/#11sRGB`

`dvipdfmx` also supports calibrated color with the `gAMA` and the `cHRM` chunk. These chunks carry information on more accurate color representation. Some software programs, however, write only `cHRM` but do not record the gamma value although the PNG specification recommends to do so. Gamma value 2.2 is assumed if only `cHRM` is present but `gAMA` is not.

Some PNG features are unavailable depending on output PDF version setting. Please refer Table 3.1 for more details.

JPEG and JPEG2000

JPEG format is supported as in `dvipdfm`. In addition to this, JPEG2000 is also supported.

JPEG and JPEG2000 image inclusion is basically done as ”pass-through”, that is, image data is not decompressed before inclusion. So, although these formats are basically lossy, there should be no quality loss when including images.

JPEG is relatively well supported. `dvipdfmx` supports embedded ICC Profiles and CMYK color. Embedded XMP metadata is also preserved in the output PDF. JFIF or Exif data might be used to determine image’s physical size.

As the PDF specification does not require information irrelevant to displaying images to be embedded, `dvipdfmx` does not embed whole data. Especially, not all application specific data is retained. Application specific data such as JFIF, Exif, and `APP14` Adobe marker are preserved. Please note that XMP and Exif data which may contain location information where the photo was taken is always preserved in the output PDF file.

JPEG2000 is also supported. It is restricted to JP2 and JPX baseline subset as required by the PDF specification. It is not well supported and still in an experimental stage. J2C format and transparency are not supported.

PDF Support

PDF inclusion is supported as in `dvipdfm`, with various important enhancement over `dvipdfm` for more robust inclusion. `dvipdfmx` can handle cross-reference streams and object streams introduced in PDF-1.5. `dvipdfmx` also supports inclusion of PDF pages other than the first page. However, tagged PDF may cause problems and annotations are not kept.

As there is no clear way to determine the natural extent of a graphics contents to be clipped, `dvipdfmx` first try to find if there is any *crop box* explicitly specified, to determine image size. If not, then it tries to refer other boundary boxes such as the *art box* which can be used for defining the extent of the page's meaningful content as suggested by the PDF Reference.[2] If there is no such page boundaries explicitly specified, useful for estimating the intended size of the graphics contents, the *media box*, which is the boundaries of the physical medium on which the page is to be printed, is used as the last resort.

The `pdf:image` special supports additional keys, "`page`" and "`pagebox`". The `page` key takes an integer value representing the page number of the PDF page to be included, and the `pagebox` takes one of the keywords `mediabox`, `cropbox`, `artbox`, `bleedbox`, or `trimbox` for selecting page's boundary box to be used. For example,

```
\special{pdf:image pagebox artbox page 3 (foo.pdf)}
```

includes 3rd page of 'foo.pdf' with the boundary box set to the art box.

Other Image Formats

For METAPOST generated Encapsulated PostScript (EPS) files, multi-byte encoding support is added. `dvipdfmx` also supports "METAPOST mode". When `dvipdfmx` is invoked with '`-M`' option, it enters in METAPOST mode and processes a METAPOST generated EPS file as an input.[1]

BMP support is also added. It is limited to uncompressed or RLE-compressed raster images. Extensions are not (won't be) supported.

For image formats not natively supported, the `-D` option can be used to convert images to PDF format before inclusion, as in `dvipdfm`. In `dvipdfmx`, the letter `v` in the `-D` option argument is expanded to the output PDF version.

3.1.2 Image Cache

Caching of images generated via filtering command specified with '`-D`' option is supported. It solves the problems that image inclusion becomes very slow when external filtering program such as GhostScript is invoked each time images are included.

Use '`-I`' option to enable this feature:

```
-I 24
```

where the integer represents the life of cache files, 24 hours in this example. Zero and negative values have a special meaning. Value 0 for "erase old cached images while leaving newly created one", −1 for "erase all cached images", and −2 for "ignore image cache". The default is −2.

[1] `prologue` should be set to `2`.

3.1.3 Custom Stream Dictionary Entries

For all supported image formats, you can insert custom entries into the image's XObject stream dictionary by enclosing the entries inside of a pair of doubled angle brackets immediately after the filename:

```
\special{pdf:image (foo.pdf) << /TestKey (TestValue) >>}
```

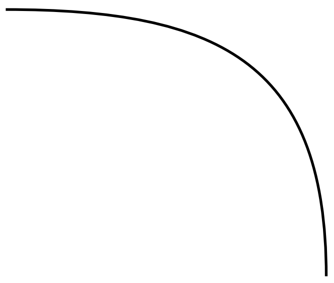
3.2 Graphics Drawing

`dvipdfmx` does not offer a high level interface to draw graphics objects. A possible way to draw graphics is to write raw PDF graphics drawing codes and then to insert them into the output via `special` commands.

To show an example, the following code:

```
\special{pdf:content
  1 0 0 1 0 0 cm
  0 100 m
  80 100 120 80 120 0 c
  S
}
```

draws a Bézier curve,



The `pdf:content` special is used here which is useful for inserting an isolated graphics object.

The above example illustrates a typical example of PDF graphics drawing. It consists of three parts; setting graphics state, constructing a path, and painting a path. A Graphic object are specified as a sequence of operators and their operands using *postfix notation. Graphics state operators* comes first, `cm` in this example sets the current transformation matrix (CTM). Then, *path construction* operators follow; move to position $(0, 100)$, append a Bézier curve from current point to $(120, 0)$ with control points $(80, 100)$ and $(120, 80)$. Finally, a *path painting* operator comes to draw the constructed path. In this example the stroking operator `S` is used.

3.2.1 The `pdf:content` Special

The `pdf:content` special can be used for drawing an *isolated* graphics object. It sets the origin of graphics drawing operators supplied to this command to the position where it is inserted.

The whole content is enclosed by a pair of graphics state save-restore operators. So for example, a color change made within a `pdf:content` command takes an effect only within the content of this special.

3.2.2 Guide to PDF Graphics Operators

PDF employs essentially the same imaging model as PostScript. So, it is easy to learn about PDF graphics drawing for people who are well accustomed to PostScript. This section is intended to be a short guide for PDF graphics operators.

Graphics State Operators

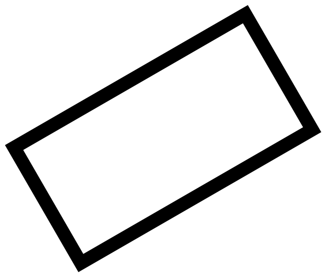
The `cm` operator modifies CTM by concatenating the specified matrix. Operands given to this operators are six numbers each representing transformation matrix elements: translation represented as $[1, 0, 0, 1, t_x, t_y]$, scaling $[s_x, 0, 0, s_y, 0, 0]$, rotation $[\cos\theta, \sin\theta, -\sin\theta, \cos\theta, 0, 0]$.

To uniformly scale the object, just use

```
2.0 0 0 2.0 0 0 cm
```

The `w` operator sets the line width, e.g., '`2 w`' sets the line width to 2. Here is an example of drawing a rotated rectangle:

```
0.866 0.5 -0.5 0.866 30 2 cm 5 w 0 0 100 50 re S
```



Transformations can be sequentially applied; for the above example,

```
1 0 0 1 30 2 cm 0.866 0.5 -0.5 0.866 0 0 cm
5 w 0 0 100 50 re S
```

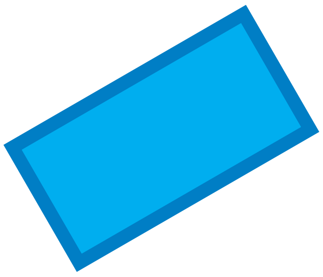
gives the same result.

To specify colors, use `RG`, `rg`, `K`, and `k` operators, for RGB or CMYK color for stroking (upper-case) and nonstroking (lower-case).

```
0.866 0.5 -0.5 0.866 30 2 cm 5 w
1 0.4 0 0 K 1 0 0 0 k
```

```
0 0 100 50 re B
```

where the B operator fill and then stroke the path.



A dash pattern can be specified with the d operator. Operands for this operator are the *dash array* and the *dash phase*. For example, to specify a dash pattern with 6-on 4-off starting with phase 0:

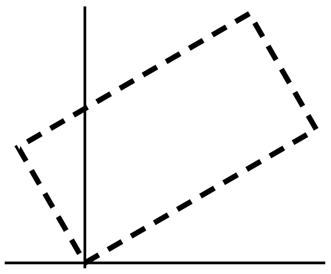
```
[6 4] 0 d 2 w 0 0 m 320 0 l S
```

draws the following dashed line:

Other important operators are q and Q, which saves and restores the graphics state.

```
1 0 0 1 30 2 cm
q
0.866 0.5 -0.5 0.866 0 0 cm
[6 4] 0 d 2 w 0 0 100 50 re S
Q
-30 0 m 90 0 l S
0 -2 m 0 96 l S
```

In the above example, d, w, and rotation only take effect within the q-Q block. The portion drawing two straight lines is unaffected by these graphics state change.



For a (incomplete) list of graphics state operators, see Talbe 3.2.

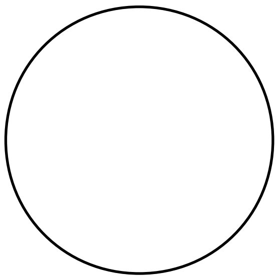
Operands	Operator	Description
—	`q`	Save the current graphics state.
—	`Q`	Restore the previously saved graphics state.
a b c d e f	`cm`	Modify the current transformation matrix by concatenating the specified matrix.
width	`w`	Set the line width.
array phase	`d`	Set the line dash pattern.
r g b	`RG`	Set the stroking color space to RGB and set the stroking color as specified.
r g b	`rg`	Set the nonstroking color space to RGB and set the nonstroking color as specified.
c m y k	`K`	Set the stroking color space to CMYK and set the stroking color as specified.
c m y k	`k`	Set the nonstroking color space to CMYK and set the nonstroking color as specified.

Table 3.2: A few examples of graphics state operators and color operators.

Path Construction Operators

A path construction normally starts with a `m` operator which moves the current point to the specified position and then sequences of other path construction operators follow. The path currently under construction is called the *current path*. A sequence of path construction operators appends segments of path to the current path and then move the *current point* to the end point of appended path. A typical sequence of path construction looks like,

```
100 50 m
100 78 78 100 50 100 c
22 100 0 78 0 50 c
0 22 22 0 50 0 c
78 0 100 22 100 50 c
S
```



This example is an approximated circle drawn by four Bézier curves.

Table 3.3 shows a list of path construction operators. Those who are accustomed to the PostScript language should note that in PDF the current path is not a part of the graphics state, and hence is *not* saved and restored along with the other graphics state parameters.

Operands	Operator	Description
x y	`m`	Begin a new path by moving the current point specified by given operands.
x y	`l`	Append a line segment from the current point to the point specified.
x_1 y_1 x_2 y_2 x_3 y_3	`c`	Append a Bézier curve to the current path. Two Control points and the end point given as operands.
x_2 y_2 x_3 y_3	`v`	Append a Bézier curve to the current path. Using the current point and first two operand as the Bézier control points.
x_1 y_1 x_3 y_3	`y`	Append a Bézier curve to the current path. The second control point coincides with the end point.
—	`h`	Close the current path by appending a straight line segment from the current point to the starting point of the path.
x y width height	`re`	Append a rectangle. First two operands for the position of lower-left corner, third and forth operand representing width and height.

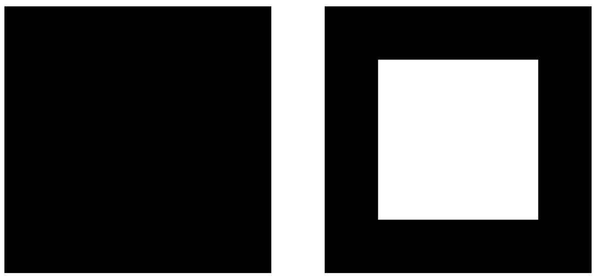
Table 3.3: List of path construction operators. All operators move the current point to the end point of appended path.

Path Painting Operators

There are basically four kind of path painting operators: `S`, `f`, `B`, and `n`. The first three for "stroke", "fill", and "fill then stroke" operators respectively, and the last one `n` paints nothing but end the path object. For filling operators, there are two kind of operators depending on how "insideness" of points are determined: the *non-zero winding number rule* and the *even-odd rule*. The even-odd rule operators are `f*` and `B*`.

The following example illustrates the difference:

```
0 0 100 100 re 20 20 60 60 re f
1 0 0 1 120 0 cm
0 0 100 100 re 20 20 60 60 re f*
```



The "interior" of the "inner" square has a non-zero even winding number. (In this example, counter-clockwise direction is assumed for both of two `re` operators.)

Chapter 4

Specials

4.1 PDF Specials

`dvipdfmx` recognizes various special commands originally introduced in `dvipdfm`. Please refer to the “Dvipdfm User’s Manual”[1] for detailed information on PDF specials.

4.1.1 Additions to PDF Specials

Several `special` commands are added for more flexible PDF generation: creation of arbitrary stream objects, controlling `dvipdfmx` behavior, and some specials which might be useful for graphics drawing.

PDF Object Manipulation

PDF object manipulation is a key feature of PDF specials. The `pdf:fstream` special is added in `dvipdfmx` which enables creation of PDF stream object from an existing file. The syntax of this special is,

```
pdf:fstream @identifier (filename) <<dictionary>>
```

where identifier and filename (specified as a PDF string object) are mandatory and a dictionary object, following the filename, which is to be added to the stream dictionary is optional.

For example, to incorporate XMP Metadata from a file `test.xmp`,

```
\special{pdf:fstream @xmp (test.xmp) <<
    /Type    /Metadata
    /Subtype /XML
>>}
\special{pdf:put @catalog << /Metadata @xmp >>}
```

Similarly, `pdf:stream` special can be used to create a PDF stream object from a PDF string instead of a file.

```
pdf:stream @identifier (stream contents) <<dictionary>>
```

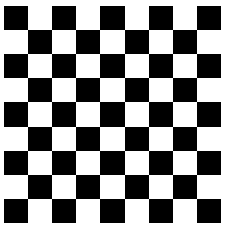


Figure 4.1: An image created by `pdf:stream` special.

This special might be useful for creating a simple image inline.

```
\special{pdf:stream @myim01
  <5500AAC05500AAC05500AAC05500AAC05500>
  <<
    /Type /XObject
    /Subtype /Image
    /BitsPerComponent 1
    /ColorSpace /DeviceGray
    /Width 9
    /Height 9
  >>
}
\special{pdf:put @resources <<
  /XObject << /MyIM01 @myim01 >>
>>}
\special{pdf:content 81 0 0 81 0 0 cm /MyIM01 Do}
```

The above example draws an image like Figure 4.1.

Controlling Font Mappings

`pdf:mapline` and `pdf:mapfile` specials can be used to append a fontmap entry or to load a fontmap file:

```
pdf:mapline foo unicode bar
pdf:mapfile foo.map
```

Specifying Output PDF Version

`pdf:majorversion` and `pdf:minorversion` specials can be used to specify major and minor version of output PDF.

```
pdf:minorversion 3
```

Please note that this command must appear on the first page, otherwise it will be ignored.

Custom File Identifiers

A custom file identifier (the `ID` entry in the trailer dictionary) can be specified via the special `pdf:trailerid`. For example: *Addition in TEX Live 2019*

```
pdf:trailerid [
  <00112233445566778899aabbccddeeff>
  <00112233445566778899aabbccddeeff>
]
```

An array of two 16-byte PDF string objects (hexadecimal notion is used in the above example) must be supplied as a file identifier. This special command must appear on the first page.

Encryption

To protect output PDF with encryption, use `pdf:encrypt` special

```
pdf:encrypt userpw (foo) ownerpw (bar) length 128 perm 20
```

where user-password (`userpw`) and owner-password (`ownerpw`) must be specified as PDF string objects. (which can be empty) Numbers specifying key-length and permission flags here are decimal numbers. See, section “Encryption Support” for a brief description of permission flags.

PDF Document Creation

As a convenience, the `pageresources` special is added, which puts given page resources into subsequent page’s *Resource Dictionary.* For example, *Addition in TEX Live 2019*

```
\special{pdf:pageresources <<
  /ExtGState << /MyGS01 << /ca 0.5 /CA 0.5 >> >>
>>}
```

puts an ExtGState resource named `MyGS01` into the current page’s and all subsequent pages’ resource dictionary.

Other notable extensions are `code`, `bcontent`, and `econtent`. The `code` special can be used to insert raw PDF graphics instructions into the output. It is different from `dvipdfm`’s `content` special in that it does not enclose contents with a `q` and `Q` (save-restore of graphics state) pair. A typical usage of this special is:

```
\special{pdf:code q 1 Tr}
...some text goes here...
\special{pdf:code Q}
```

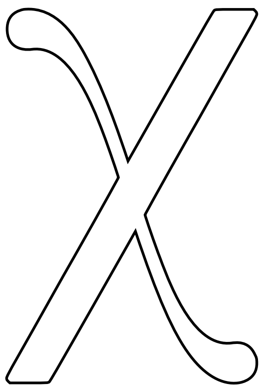


Figure 4.2: A character drawn in the PDF text rendering mode 1.

which changes text rendering mode to 1, as shown in Figure 4.2.

Be careful on using this special as it is very easy to generate broken PDF files. The `bcontent` and `econtent` pair is somewhat fragile and might be incompatible to other groups of special commands. It may not always be guaranteed to work as 'expected'.

4.1.2 ToUnicode Special

PDF allows users to attach various additional information such as document information, annotation, and navigation information (like bookmarks) to their document. All human-readable text, *text string*, contained in such information must be encoded either in *PDFDocEncoding* or UTF-16BE with Unicode byte order marker. However, as many users can't prepare their document with text strings properly encoded, there is a special kind of special command, `pdf:tounicode`, which can be used to convert text strings into the appropriate Unicode form. Note that this feature is provided just as a remedy for users incapable of encoding text strings properly.

For example,

```
\special{pdf:tounicode 90ms-RKSJ-UCS2}
```

declares that *some* of text strings should be re-encoded according to specified conversion CMap `90ms-RKSJ-UCS2`.

Conversion is done only for arguments to several PDF special commands such as `docinfo`, `ann`, and `out` but not for that of general object creation specials. Note that not all PDF string objects are subject to this conversion. By default, only dictionary entries listed below are converted.

```
Title Author Subject Keywords Creator Producer Contents Subj
TU T TM
```

The list of dictionary entries subject to conversion can be extended by supplying additional dictionary keys as an array object:

Addition in TEX Live 2019

```
\special{pdf:tounicode 90ms-RKSJ-UCS2 [/RC /DS]}
```

If the name of conversion CMap contains one of the keywords RKSJ, B5, GBK, and KSC, PDF string objects are treated specially when they are parsed. A two byte sequence starting with the first byte's high bit set is treated "as is" so that the `0x5c` byte appears in the second byte is not treated as an escape sequence. This behavior is not compliant to the PDF specification.

4.1.3 PDF Special Examples

This section shows several examples of special command usage. It is intended to be a hint for advanced users, so uninterested users can safely skip this section. In many cases, using `dvipdfmx` PDF specials requires knowledge on PDF. Please refer to Adobe's "PDF Reference"[2].

Annotations

In this section, some useful special commands for creating *annotations* are explained. Note that viewer support is required for annotations to be displayed correctly.

First start with a very simple *Text Annotation* for attaching a comment. This feature is supported by many PDF viewer applications.

```
\special{pdf:ann width 20bp height 20bp
  <<
     /Type     /Annot
     /Subtype  /Text
     /Name     /Comment
     /T        (Text Annotation Example)
     /Subj     (An Example of Text Annotations)
     /Contents (A Quick Brown Fox Jumped Over The Lazy Dog.)
  >>
}
```

`pdf:ann` special is used to create an annotation. A small icon shall be shown on the side margin. Here, dictionary entry `T` is for the tilte, `Subj` for the subject of this annotation, and `Contents` for the text string to be shown in the body of this annotation.

Likewise, *Rubber Stamp Annotation*, which places a rubber stamp like figure,

```
\special{pdf:ann width 150bp height 50bp
  <<
     /Type     /Annot
     /Subtype  /Stamp
     /Name     /Approved
  >>
}
```

Other keywords such as `Expired`, `Final`, `Draft`, and so on, can be used in place of `Approved`.

One can create stamps of their own style. For this purpose, other specials `pdf:bxobj` and `pdf:exobj` can be used for designing stamps. Those specials “capture” all typeset material enclosed by them into a PDF *Form XObject*, which is a reusable graphics object like included images.

For a simple example,

```
\special{pdf:bxobj @MyStamp
             width 280pt height 0pt depth 40pt}
\addfontfeature{Scale=3,Color=FF9933}My Own Stamp%
\special{pdf:exobj}
```

It captures typeset material “My Own Stamp” (this example uses `fontspec` package’s command for changing font size and text color) into the object labeled as `MyStamp` for later reuse. Then, `AP` (*appearance dictionary*) entry for controlling the appearance of annotations is used as,

```
\special{pdf:ann width 280pt height 40pt
  <<
    /Type    /Annot
    /Subtype /Stamp
    /AP      << /N @MyStamp >>
  >>
}
```

The image captured into the object `MyStamp` is used as “Normal” (`AP` dictionary entry `N`) appearance. (`R` for “Rollover” and `D` for “Down” can be used.)

The result is:

My Own Stamp

With the following code, `dvipdfmx` reads the source file and creates a stream object named `SourceFile`, and then creates file attachment annotation.

```
\special{pdf:fstream @SourceFile (\jobname.tex)}%
\special{pdf:ann width 10bp height 20bp
   <<
      /Type /Annot
      /Subtype /FileAttachment
      /FS <<
            /Type /Filespec
            /F    (\jobname.tex)
            /EF   << /F @SourceFile >>
          >>
      /Name     /PushPin
      /C        [0.8 0.2 0.2]
      /T        (The Dvipdfmx User's Manual)
      /Subj     (The Dvipdfmx User's Manual)
      /Contents (XeLaTeX source file of the manual.)
   >>
}
```

A push-pin image must be shown on the margin if viewer supports this kind of annotation. PDF viewer applications are required to provide predefined icon appearances at least for the following standard icons: `Graph`, `PushPin`, `PaperClip`, and `Tag`.

Special Color Space

This section shows various examples of using *Special color spaces*. Examples in this section have a common structure. They consist of essentially three parts. The first part is for defining color space itself. PDF object creation commands like `pdf:obj` and `pdf:stream` are used for this purpose. Next is for registering color space resources in the page's *Resource Dictionary*. It can be done via `pdf:put` command as,

```
\special{pdf:put @resource <<
  /Category << ...key-value pairs... >>
>>}
```

where `@resource` is a special keyword representing current page's Resource Dictionary and `Category` (to be replaced by actual category name) is a category name such as `ColorSpace`. Finally, graphics objects are placed, with or without a combination of text and, PDF drawing operators inserted by the `pdf:code` or the `pdf:contents` special.

The first example is the *Separation* color space:

Orange and Green

```
\special{pdf:stream @TintTransform1
  ({0 exch dup 0.62 mul exch 0})
  <<  /FunctionType 4
      /Domain [ 0.0 1.0 ]
      /Range  [ 0.0 1.0 0.0 1.0 0.0 1.0 0.0 1.0 ]
  >>
}
\special{pdf:stream @TintTransform2
  ({dup 0.78 mul exch dup 0.05 mul exch 0.71 mul 0})
  <<  /FunctionType 4
```

```
        /Domain [ 0.0 1.0 ]
        /Range  [ 0.0 1.0 0.0 1.0 0.0 1.0 0.0 1.0 ]
    >>
  }
  \special{pdf:obj @Orange [
      /Separation /Orange /DeviceCMYK @TintTransform1
    ]
  }
  \special{pdf:obj @Green [
      /Separation /Green /DeviceCMYK @TintTransform2
    ]
  }
  \mbox{%
    \special{pdf:put @resources <<
        /ColorSpace << /CS01 @Orange /CS02 @Green >>
      >>
    }%
    \special{pdf:code q /CS01 cs 1.0 scn}
    Orange
    \special{pdf:code Q}
    and
    \special{pdf:code q /CS02 cs 1.0 scn}
    Green
    \special{pdf:code Q}
  }
```

`TintTransform`'s defined here are functions for transforming *tint* values into approximate colors in the *alternate color space* (`DeviceCMYK` in this example). PostScript calculator functions are used for converting a single component value representing "Orange" or "Green" into four component CMYK values to approximate those colors. The "Orange" color v is approximated as $(0, 0.62v, v, 0)$ in CMYK color space for alternate display here.

The `cs` operator for selecting color space and the `scn` operator for color values are used in the `pdf:code` special. Be sure that the `pdf:put` command, which puts color space resources into the current page's Resource Dictionary, goes into the same page as subsequent drawing commands.

`dvipdfmx` currently does not have an easy interface for using various color space families such as CIE-Based color spaces (e.g., calibrated colors and color space with an ICC profile) and Special color spaces (e.g., indexed, separation, shading and patterns).

Another example is a *shading pattern*:

```
  \special{pdf:put @resources <<
    /Shading <<
      /SH01 <<
        /ShadingType 2
        /ColorSpace @Orange
        /Coords     [0 0 320 20]
        /Extend     [true true]
        /Function   << /FunctionType 2 /Domain [0 1] /N 1.0 >>
      >>
    >>
  >>}
  \special{pdf:content 0 0 320 20 re W n /SH01 sh}
```

where the "Orange" separation color space defined before is used again. This example shows an axial shading (`ShadingType` 2) pattern.

The shading pattern requires coordinate values to be mapped into color values. Type 2 (Exponential Interpolation) *Function* is used for defining how this mapping should occur here. The above example, with the exponent $N = 1$, is just a simple linear-gradient.

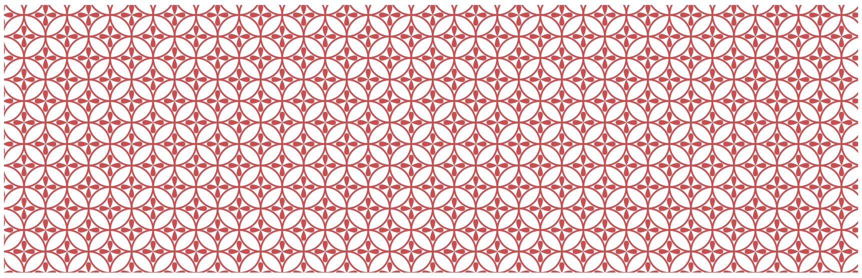
The final examples is a *tiling pattern*.

```
\special{pdf:stream @MyPattern
    (0.16 0 0 0.16 0 0 cm 4 w
     50 0 m 50 28 28 50 0 50 c S 100 50
     m 72 50 50 28 50 0 c S
     50 100 m 50 72 72 50 100 50 c S
     0 50 m 28 50 50 72 50 100 c S
     100 50 m 100 78 78 100 50 100 c 22 100 0 78 0 50 c
     0 22 22 0 50 0 c 78 0 100 22 100 50 c S
     0 0 m 20 10 25 5 25 0 c f 0 0 m 10 20 5 25 0 25 c f
     100 0 m 80 10 75 5 75 0 c f
     100 0 m 90 20 95 25 100 25 c f
     100 100 m 80 90 75 95 75 100 c f
     100 100 m 90 80 95 75 100 75 c f
     0 100 m 20 90 25 95 25 100 c f
     0 100 m 10 80 5 75 0 75 c f
     50 50 m 70 60 75 55 75 50 c 75 45 70 40 50 50 c f
     50 50 m 60 70 55 75 50 75 c 45 75 40 70 50 50 c f
     50 50 m 30 60 25 55 25 50 c
     25 45 30 40 50 50 c f
     50 50 m 60 30 55 25 50 25 c 45 25 40 30 50 50 c f)
    <<
        /BBox [0 0 16 16]
        /PaintType 2
        /PatternType 1
        /Resources <<
            /ProcSet [/PDF]
        >>
        /TilingType 3
        /Type /Pattern
        /XStep 16
        /YStep 16
    >>
}
```

The above example defines a tiling pattern. The content stream containing painting operators, `m` for “move-to”, `c` for “curve-to”, `f` for “fill”, and `S` for “stroke”, defines the appearance of the *pattern cell* for this tiling pattern. With the following code,

```
\special{pdf:put @resources
    <<
        /ColorSpace << /CS01 [/Pattern /DeviceRGB] >>
        /Pattern << /PT01 @MyPattern >>
    >>
}
\special{pdf:content
    q 0.8 0.3 0.3 RG /CS01 cs 0.8 0.3 0.3 /PT01 scn
    0 0 320 100 re f
}
```

a box filled with the tiling pattern defined above is drawn.



Transparency

XƎLATEX’s transparency feature is currently lost in `xdvipdfmx`, but the same effect can be achieved by setting graphics state parameters with `ExtGState` resources and `gs` operator. Here is a simple transparency example:

```
\special{pdf:obj @GS01 <<
  /Type /ExtGState /CA 0.5 /ca 0.5 /AIS false
>>}%
\mbox{%
  \special{pdf:put @resources <<
    /ExtGState << /GS01 @GS01 >>
  >>}%
  \special{pdf:code q /GS01 gs 1.0 0.8 0.2 rg}%
  α%
  \special{pdf:code 0.4 0.8 0.4 rg}%
  \hspace{-0.3em}%
  β%
  \hspace{-0.3em}\raisebox{0.5ex}{%
```

```
    \special{pdf:code 0.4 0.4 0.8 rg}%
    π%
  }%
  \special{pdf:code 1.0 0.2 0.4 rg}%
  \hspace{-0.2em}%
  γ%
  \special{pdf:code Q}%
}
```

where values for `CA` and `ca` represent opacity of stroke and fill color respectively. Again, `pdf:put` command must go into the same page as subsequent graphics and text drawing operators.

4.2 Dvipdfmx Extensions

A new special `dvipdfmx:config` was introduced in TEXLive 2016 which makes it possible to invoke a command line option. Several single letter command line options except '`D`' are supported. For example,

Addition in TEX Live 2016

```
dvipdfmx:config C 0x10
```

sets the `dvipdfmx`'s compatibility flags. See, the section "Incompatible Changes" for an explanation of compatibility flags.

4.3 PS Specials

PS (PostScript) specials can be used to insert a raw PostScript code for drawing graphics objects and transforming subsequent text and graphics. Please note that support for PostScript operators in `dvipdfmx` is very limited. It is just enough for interpreting PostScript figures output by METAPOST. Only a basic set of operators for arithmetic and math, stack operation and

Classification	Operators
Arithmetic & Math	`add sub mul div neg truncate`
Stack Operation	`clear pop exch`
Graphis State	`gsave grestore setlinewidth setdash setlinecap setlinejoin setmiterlimit setgray setrgbcolor setcmykcolor`
Coordinate System	`concat scale translate rotate idtransform dtransform`
Path Construction	`currentpoint newpath closepath moveto rmoveto lineto rlineto curveto rcurveto arc arcn clip eoclip`
Painting	`stroke fill`
Glyph & Font	`show findfont scalefont setfont currentfont stringwidth`

Table 4.1: List of PostScript operators recognized by `dvipdfmx`.

manipulation, graphics state, path construction and painting, glyph and font, are supported. See, Table 4.1 for the list of recognized PostScript operators.

It might be enough for the purpose of basic graphics drawings but as there are no support for conditionals and controls it is not enough for complicated tasks, especially, the PSTricks package is not supported.

In `dvipdfmx`, text handling is extended to support CJK text. The following code draws Japanese text like shown in Figure 1.1:

```
\special{pdf:mapline urml UniJIS-UTF8-H ipaexm.ttf}
\special{ps: urml findfont 16 scalefont setfont
  currentpoint moveto
  (...some Japanese text goes here...) show
}
```

Chapter 5

Fonts and Encodings

5.1 Fonts and Encodings Support

In `dvipdfmx`, all font formats supported by `dvipdfm` are also supported with many improvements: The CFF conversion for PostScript Type1 fonts[1] is implemented which greatly reduces the output file size. Embedded TrueType fonts are now subsetted. The OpenType font format is also supported.[2]

There are various enhancements made for supporting Unicode and legacy multi-byte character encodings for East Asian languages.

5.2 Font Mappings

The Syntax of font-mapping (fontmap) files is basically the same as in `dvipdfm`. There are few extensions available in `dvipdfmx`. In addition to the 8-bit `enc` file and keywords `builtin` and `none`, `dvipdfmx` accepts a PostScript CMap Resource name and the keyword `unicode` in the encoding field.

When the keyword `unicode` is specified in the encoding field of fontmap files, it is assumed that Unicode values are used in the input DVI file.

```
urml    unicode  SourceHanSerifJP-Light.otf
```

Although the DVI format allows 3-byte and 4-byte character codes to be used, `dvipdfmx` only supports up to 2-byte range since there is no TFM format supporting 3-byte or 4-byte codes.

For PostScript Type1 fonts which do not support Unicode natively, an auxiliary file, the Adobe Glyph List, is required to make it possible to use fonts with Unicode access.

As a general framework for supporting legacy multi-byte encodings, `dvipdfmx` employs PostScript CMap Resources for handling input strings encoded in various character encodings. A CMap name can be specified in the encoding field just like the encoding name for 8-bit encodings. For example, to specify the CMap “UniJIS-UCS2-H”,

[1]PostScript Type1 font support is restricted to the binary format as in `dvipdfm`.

[2]Its implementation is based on the OpenType specification version 1.4. Newly added features such as color fonts and variable fonts are not supported yet.

```
urml    UniJIS-UCS2-H  HiraMinPro-W3.otf
```

For information on the Adobe Glyph List and PostScript CMap Resources, see, the section 2, "Auxiliary Files".

5.2.1 Extended Syntax and Options

Few options are available in `dvipdfmx` in addition to the original dvipdfm's one. Please note that all features which makes `dvipdfmx` to use non-embedded fonts are deprecated, as by doing so it makes `dvipdfmx` to create PDF files which can be non-compliant to the ISO standards.

SFD Specification

For bundling up a font split into multiple subfonts via SFD back into a single font, dvipdfmx supports extended syntax of the form

```
tfm_name@SFD@  encoding  filename  options
```

A typical example looks like:

```
gbsn@EUC@  GB-EUC-H  gbsn00lp
```

where TFMs `gbsn00`, `gbsn01`, `gbsn02`... are mapped into a single font named `gbsn00lp` via the rule described in the SFD file `EUC`.

TrueType Collection Index

TrueType Collection index number can be specified with `:n:` in front of the TrueType font name:

```
min10  H  :1:mincho
```

In this example, the option `:1:` tells `dvipdfmx` to select first TrueType font from the TTC font `mincho.ttc`. Alternatively, the '`-i`' option can be used in the option field to specify TTC index:

```
min10  H  mincho  -i 1
```

Non-embedding Switch

The character '`!`' in front of the font name can be used to indicate that the font shall not be embedded. This feature greatly reduces the size of the final PDF output, but the PDF file may not be viewed exactly the same in other systems on which appropriate fonts are not installed.

Use of this option is deprecated.

Character Collection	Font Family	Description
Adobe-Japan1	Ryumin-Light	PS printers
	GothicBBB-Medium	
Adobe-CNS1	MHei-Medium-Acro	Acrobat Reader 4
	MSung-Light-Acro	
Adobe-GB1	STSong-Light-Acro	
	STHeiti-Regular-Acro	
Adobe-Japan1	HeiseiMin-W3-Acro	
	HeiseiKakuGO-W5-Acro	
Adobe-Korea1	HYGoThic-Medium-Acro	
	HYSMyeongJo-Medium-Acro	
Adobe-CNS1	MSungStd-Light-Acro	Acrobat Reader 5
Adobe-GB1	STSongStd-Light-Acro	
Adobe-Korea1	HYSMyeongJoStd-Medium-Acro	
Adobe-CNS1	AdobeMingStd-Light-Acro	Adobe Reader 6
Adobe-GB1	AdobeSongStd-Light-Acro	
Adobe-Japan1	KozMinPro-Regular-Acro	
	KozGoPro-Medium-Acro	
Adobe-Korea1	AdobeMyungjoStd-Medium-Acro	
Adobe-CNS1	AdobeHeitiStd-Regular	Adobe Reader 7
Adobe-Japan1	KozMinProVI-Regular	Adobe Reader 8

Table 5.1: List of available 'Standard' CJK font. Most of them are available as a part of Adobe Asian Font Packs for each versions of Adobe or Acrobat Reader.

NOTE: `dvipdfmx` always converts input encodings to CIDs and then uses Identity CMaps[3] in the output PDF. However, ISO 32000-1:2008 describes as

> *The Identity-H and Identity-V CMaps shall not be used with a non-embedded font. Only standardized character sets may be used.*

which had never appeared in Adobe's PDF References. This makes all PDF files generated by `dvipdfmx` with non-embedded CID-keyed fonts non-compliant to the ISO standards.

'Standard' CJK Fonts

Use of this feature shall be avoided for new documents. It is described here since it might still be useful for some situations.

This feature is deprecated.

`dvipdfmx` recognizes several 'Standard' CJK fonts although there are no such notion in PDF. In older days where there were not so many freely available CJK fonts, it was sometimes useful to create PDF files without embedding fonts and let PDF viewers or printers to use substitute fonts (tend to be higher quality) installed in their systems. `dvipdfmx` 'knows' several fonts which might be available in PostScript printers and PDF applications such as Acrobat Reader, and uses them without actually having it. See, Table 5.1, for the list of available 'Standard' CJK fonts.

Only fixed-pitch glyphs (i.e., quarter, third, half, and full widths) are supported for those fonts.

[3]Predefined CMaps `Identity-H` and `Identity-V` for the identity mapping.

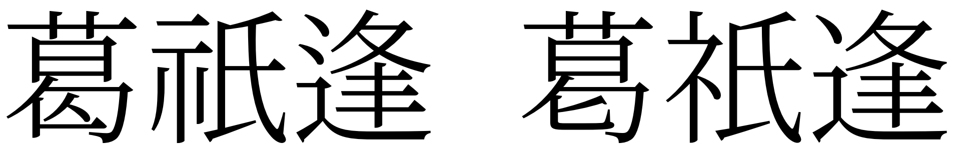


Figure 5.1: JIS2004 vs. JIS1990 form.

Stylistic Variants

Keywords `,Bold`, `,Italic`, and `,BoldItalic` can be used to create synthetic bold, italic, and bolditalic style variants from other font using PDF viewer's (or OS's) function.

Use of this option is deprecated.

```
jbtmo@UKS@  UniKSCms-UCS2-H  :0:!batang,Italic
```

Availability of this feature highly depends on the implementation of PDF viewers. This feature is usually not supported for embedded fonts. Notice that this option automatically disables font embedding thus use of it is deprecated.

5.2.2 Specifying Unicode Plane

As there is no existing TFM format supporting 3-byte or 4-byte character encodings, the only way to use Unicode characters other than the BMP is to map the code range 0-65535 to different planes via (e.g., to plane 1) the '`-p 1`' fontmap option. This option is available only when `unicode` is specified in the encoding field.

5.2.3 OpenType Layout Feature

The OpenType Layout Feature fontmap options mentioned below are only meaningful when `unicode` is specified in the encoding field.

With the '`-w`' option, writing mode can be specified. '`-w 1`' denotes the font is for vertical writing. It automatically enables an OpenType Layout Feature related to vertical writing, namely, `vert` or `vrt2`, to choose proper glyphs for vertical text.

The '`-l`' (lower case el) option can be used to enable various OpenType Layout GSUB Features. For example, '`-l jp04`' enables `jp04` feature to select JIS2004 forms for Kanjis. Features can be specified as a ":" separated list of OpenType Layout Feature tags like '`-l vkna:jp04`'. Script and language may be additionally specified as '`-l kana.JAN.ruby`'.

Addition in TEX Live 2017.

An example can be

```
uprml-v  unicode  SourceHanSerifJP-Light.otf  -w 1 -l jp90
```

which declares that font should be treated as for vertical writing and use JIS1990 form for Kanjis.

This feature is limited to the single substitution, there are no way to select a glyph from multiple candidates, such as in the `aalt` feature, and specifying general many-to-many glyph substitutions does not take effect.

5.3 Other Improvements

This section briefly describes other improvements made for `dvipdfmx`. There is an extension to glyph name handling in the `enc` file for seamless support of both PostScript Type1 and TrueType fonts. PostScript Type1 font support is enhanced although this format might be considered obsolete.

5.3.1 Extended Glyph Name Syntax

`dvipdfmx` accepts the following syntax for glyph names in the `enc` file: `uni0130`, `zero.onum` and `T_h.liga`. Each represents a glyph accessed with Unicode value `U+0130`, oldstyle number for zero and “Th” ligature accessed via the OpenType Layout GSUB Feature `onum` and `liga`, respectively. Note that `dvipdfmx` does not understand glyph names which directly use a glyph index such as `index0102` or `gid2104`.

When `dvipdfmx` encounters a glyph name, e.g., `T_h.liga`, it first looks for OpenType `post` table if such glyph name exists; if it exists, then `dvipdfmx` simply uses `post` table and maps the glyph name to the glyph index; if not, `dvipdfmx` tries to convert `T_h` to a Unicode sequence (U+0054 U+0068 in this example) via the AGL mapping; then, OpenType `cmap` table is used to further convert the resulting Unicode sequence to the sequence of glyph indices; finally, the OpenType Layout Feature `liga` is applied to get the desired glyph.

A glyph name of a form `a.swsh2` can be specified to denote the 2nd swash variant form of the letter ‘a’.

5.3.2 CFF Conversion

`dvipdfmx` supports on-the-fly PostScript Type1 to CFF (Type1C) conversion which greatly reduces size of the resulting PDF file when using Type1 fonts. Conversion is essentially ‘lossless’ and there should not be any quality loss. However, due to differences in the ability of rasterizers, there might be noticeable differences on rendering result.

When an older Type1 font is used, `dvipdfmx` may give the following warning message:

```
Obsolete four arguments of "endchar" will be used for Type1
"seac" operator.
```

It happens when there is an accented character made as a composite glyph using the “seac” operator. This warning is issued as conversion can’t be done without relying on the *deprecated* usage of the `endchar` operator. However, as mentioned in “Appendix C Compatibility and Deprecated Operators” of Adobe Technical Note #5177, “Type 2 Charstring Format”, PDF applications should support this operator and hence this warning message can be ignored.

Use of Type1 font should be avoided as much as possible. Please consider using OpenType version instead.

5.4 Font Licensing

In OpenType font format, information regarding how a font should be treated when creating a document can be recorded.[4]. `dvipdfmx` uses this information to decide whether font embedding is permitted.

[4]See, “OpenType Specification: OS/2 – OS/2 and Windows Metrics Table”

This font licensing information is indicated by the flag called `fsType` recorded in OpenType font files; each bits representing different restrictions on font embedding. If multiple flag bits are set in `fsType`, the least restrictive license granted takes precedence in `dvipdfmx`. The `fsType` flag bit recognized by `dvipdfmx` is as follows:

- Installable embedding
- Editable embedding
- Embedding for Preview & Print only

`dvipdfmx` issues the following warning message for fonts with 'Preview & Print only' setting:

```
This document contains 'Preview & Print' only licensed font
```

For a font with this type of licensing, font embedding is allowed solely for the purpose of (on-screen) viewing and/or printing; further editing of the document or extracting embedded font data for other purposes are not allowed. One way to ensure this condition is to protect your document with a non-empty password.

All other flags are treated as more restrictive license than any of the above flags and treated as "No embedding allowed"; e.g., if both of the editable-embedding flag and unrecognized license flag is set, the font is treated as editable-embedding allowed, however, if only unrecognized flags are set, the font is not embedded.

Font Embedding flags are preserved in the embedded font if they are embedded as a TrueType font or a CIDFontType2 CID-keyed font. For all fonts embedded as a PostScript font (Type1C and CIDFontType0 CID-keyed font), they are not preserved. Only `Copyright` and `Notice` in the *FontInfo* dictionary are preserved in this case.

Some font vendors put different embedding restrictions for different condition; e.g., font embedding might not be permitted for the commercial use unless you acquire the "commercial license" separately. Please read EULA carefully before making decision on the font usage.

See, for example, Adobe's site on font embedding permissions for the font in the Adobe Type Library. Microsoft also has a FAQ page on Font Redistribution.

For Japanese fonts in general, embedding permission tend to be somewhat restrictive. Japanese users should read the statement regarding font embedding from Japan Typography Association (in Japanese):

`https://www.typography.or.jp/act/morals/moral4.html`

`dvipdfmx` does not support full embedding. Only subset embedding is supported.

Chapter 6

Encryption

6.1 Encryption Support

`dvipdfmx` offers basic PDF password security support including the 256-bit AES encryption. Only the "Standard" security handler is supported and the Public-key security handler is not supported.

Encryption is enabled by '`-S`' command line option.

```
dvipdfmx -S -K 128 -P 0x14
```

where `-K` and `-P` options are used to specify encryption key length and permission flags respectively, and are briefly explained in Table 6.1.

When `dvipdfmx` is invoked with encryption via the `-S` option, passwords will be asked. However, in some circumstances, it might be desirable to avoid being prompted for passwords. In that case, use the `pdf:encrypt` special to supply passwords in the TEX file, as,

```
\special{pdf:encrypt userpw (foo) ownerpw (bar) length 128 perm 20}
```

Here, user and owner passwords are supplied as PDF string objects (`foo` and `bar` in the example above) which can be empty.

Up to two passwords can be specified for a document – an owner password and a user password. If a user attempts to open an encrypted document with user password being set,

Option	Description
`-S`	Enable PDF encryption.
`-K` *number*	Set encryption key length. The default value is 40.
`-P` *number*	Set permission flags for PDF encryption. The *number* is a 32-bit unsigned integer representing permission flags. See, Table 6.2. The default value is `0x003C`.

Table 6.1: Command line options for encryption.

Bit Position	Meaning
3	(Revision 2) Print the document. (Revision 3 or greater) Print the document. Print quality depending on bit 12.
4	Modify the contents of the document by operations other than those controlled by bits 6, 9, and 11.
5	Copy or extract text and graphics from the document.
6	Add or modify text annotations, fill in interactive form fields. Creation and modification of interactive form field is also allowed if bit 4 is set.
9	(Revision 3 or greater) Fill in existing interactive form fields (including signature fields), even if bit 6 is clear.
10	*Deprecated in PDF 2.0* (Revision 3 or greater) Extract text and graphics (in support of accessibility to users with disabilities or for other purposes).
11	(Revision 3 or greater) Assemble the document (insert, rotate, or delete pages and create document outline items or thumbnail images), even if bit 4 is clear.
12	(Revision 3 or greater) High-quality printing. When this bit is clear (and bit 3 is set), printing shall be limited to a low-level, possibly of degraded quality.

Table 6.2: Flag bits and brief explanations. Revision 2 is used when the encryption key length is 40 bits or when PDF output version is less than 1.5. Otherwise, Revision 3 or greater is used.

PDF application should prompt for a password. Users are allowed to access the contents of the document only when either password is correctly supplied. Depending on which password (user or owner) was supplied, additional operations allowed for the opened document is determined; full access for users who opened with the correct owner password or additional operations controlled by permission flags for users who opened with the correct user password.

Although PDF specification allows various character encodings other than `US-ASCII` for entering password, `dvipdfmx` is unable to handle it properly. Thus it must be assumed that `US-ASCII` is used for password strings.

Access permission flags can be specified via '`-P`' command-line option. Each bits of (32-bit unsigned) integer number given to this option represents user access permissions; e.g., bit position 3 for allowing "print", 4 for "modify", 5 for "copy or extract", and so on. See, Table 6.2. For example, `-P 0x34` allows printing, copying and extraction of text, and adding and modifying text annotation and filling in interactive form fields (but disallows modification of the contents of the document).

The '`-K`' option can be used to specify the encryption key length. The key length must be multiple of 8 in the range 40 to 128, or 256 (for PDF version 1.7 plus Adobe Extension or PDF version 2.0). Please note that when key length 256 is specified for PDF version 1.7 output, it requires Adobe's Extension to PDF-1.7 and hence PDF applications may not support it. PDF version 1.4 is required for key length more than 40 bits. AES encryption algorithm requires PDF version 1.6.

To show some examples:
128-bit AES encryption with print-only (high-quality) setting,

```
dvipdfmx -V 5 -S -K 128 -P 0x804 input.dvi
```

256-bit AES encryption with print (low-quality), adding and modifying text annotations allowed,

```
dvipdfmx -V 2.0 -S -K 256 -P 0x24 input.dvi
```

The default value for '`-K`' is 40 and for '`-P`' is `0x003C0` (all bits from bit-position 3 to 6 set).

Chapter 7

Compatibility

7.1 Incompatible Changes

There are various minor incompatible changes to `dvipdfm`.

The '`-C`' command line option may be used for compatibility to `dvipdfm` or older versions of `dvipdfmx`. The '`-C`' option takes flags meaning

- bit position 2: Use semi-transparent filling for tpic shading command, instead of opaque gray color. (requires PDF 1.4)
- bit position 3: Treat all CID-keyed font as fixed-pitch font. This is only for compatibility.
- bit position 4: Do not replace duplicate fontmap entries. `dvipdfm` behavior.
- bit position 5: Do not optimize PDF destinations. Use this if you want to refer from other files to destinations in the current file.
- bit position 6: Do not use predictor filter for Flate compression.
- bit position 7: Do not use object stream.

The remap option '`-r`' in fontmaps is no longer supported and is silently ignored. The command line option '`-e`' to disable partial (subset) font embedding is not supported.

7.2 Important Changes

Here is a list of important changes since the TEX Live 2016 release:

- Changes to make PDF/A creation easier: Always write CIDSet and CharSet for embedded fonts. Do not compress XMP metadata.
- Merge from libdpx for pTEX-ng by Clerk Ma.
- Addition of `STHeiti-Regular-Acro` for CJK 'Standard' fonts.
- Command line option '`-p`' takes precedence over `papersize` and `pagesize` specials.
- Fixed serious bugs in supporting '`unicode`' encoding: OpenType Layout Feature `vert` and `vrt2` was not enabled. Support for format 2 CFF charsets was broken.
- Added simplified version of OpenType Layout support: The "`-l`" option in fontmaps.

The full `ChangeLog` entries can be viewed via the web interface of the TEX Live SVN repository:

`https://tug.org/svn/texlive/trunk/Build/source/texk/dvipdfm-x`

There was an undocumented feature for supporting OpenType Layout but it was dropped. Simplified support for the OpenType Layout was introduced instead.

Further Reading

[1] "Dvipdfm User's Manual" written by Mark A. Wicks.

[2] Adobe's PDF References and a free copy of ISO 32000-1:2008 standard are available from "PDF Technology Center" on Adobe Developer Connection.

[3] The OpenType Specification is available from Microsoft's site: "OpenType Specification".

[4] "Portable Network Graphics (PNG) Specification (Second Edition)".

[5] An article regarding DVI specials: Jin-Hwan Cho, "DVI specials for PDF generation", TUGboat 30(1):6–11, 2009.

Appendix A

GNU Free Documentation License v1.3

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "**Document**", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "**you**". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "**Modified Version**" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "**Secondary Section**" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "**Invariant Sections**" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "**Cover Texts**" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "**Transparent**" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "**Opaque**".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "**Title Page**" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "**publisher**" means any person or entity that distributes copies of the Document to the public.

A section "**Entitled XYZ**" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "**Acknowledgements**", "**Dedications**", "**Endorsements**", or "**History**".) To "**Preserve the Title**" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See `https://www.gnu.org/licenses/`.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy’s public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

“Massive Multiauthor Collaboration Site” (or “MMC Site”) means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that