WIKIPEDIA

PDF

The **Portable Document Format** (**PDF**) is a file format developed in the 1990s to present documents, including text formatting and images, in a manner independent of application software, hardware, and operating systems.[3][4] Based on the PostScript language, each PDF file encapsulates a complete description of a fixed-layout flat document, including the text, fonts, vector graphics, raster images and other information needed to display it. PDF was standardized as an open format, ISO 32000, in 2008, and does not require any royalties for its implementation.

Today, PDF files may contain a variety of content besides flat text and graphics including logical structuring elements, interactive elements such as annotations and form-fields, layers, rich media (including video content) and three dimensional objects using U3D or PRC, and various other data formats. The PDF specification also provides for encryption and digital signatures, file attachments and metadata to enable workflows requiring these features.

Portable Document Format

Adobe PDF icon

Filename extension	`.pdf`[1]
Internet media type	`application/pdf`,[2] `application/x-pdf` `application/x-bzpdf` `application/x-gzpdf`
Type code	'PDF '[2] (including a single space)
Uniform Type Identifier (UTI)	com.adobe.pdf
Magic number	%PDF
Developed by	ISO
Initial release	15 June 1993
Latest release	2.0
Extended to	PDF/A, PDF/E, PDF/UA, PDF/VT, PDF/X
Standard	ISO 32000-2
Open format?	Yes
Website	www.iso.org/standard/63534.html (https://www.iso.org/standard/63534.html)

Contents

History and standardization

Adobe Systems made the PDF specification available free of charge in 1993. In the early years PDF was popular mainly in desktop publishing workflows, and competed with a variety of formats such as DjVu, Envoy, Common Ground Digital Paper, Farallon Replica and even Adobe's own PostScript format.

PDF was a proprietary format controlled by Adobe until it was released as an open standard on July 1, 2008, and published by the International Organization for Standardization as ISO 32000-1:2008,[5][6] at which time control of the specification passed to an ISO Committee of volunteer industry experts. In 2008, Adobe published a Public Patent License to ISO 32000-1 granting royalty-free rights for all patents owned by Adobe that are necessary to make, use, sell, and distribute PDF compliant implementations.[7]

PDF 1.7, the sixth edition of the PDF specification that became ISO 32000-1, includes some proprietary technologies defined only by Adobe, such as Adobe XML Forms Architecture (XFA) and JavaScript extension for Acrobat, which are referenced by ISO 32000-1 as normative and indispensable for the full implementation of the ISO 32000-1 specification. These proprietary technologies are not standardized and their specification is published only on Adobe's website.[8][9][10][11][12] Many of them are also not supported by popular third-party implementations of PDF.

On July 28, 2017, ISO 32000-2 (PDF 2.0) was published by the ISO. ISO 32000-2 does not include any proprietary technologies as normative references.[13]

Technical foundations

The PDF combines three technologies:

- A subset of the PostScript page description programming language, for generating the layout and graphics.
- A font-embedding/replacement system to allow fonts to travel with the documents.
- A structured storage system to bundle these elements and any associated content into a single file, with data compression where appropriate.

PostScript

PostScript is a page description language run in an interpreter to generate an image, a process requiring many resources. It can handle graphics and standard features of programming languages such as `if` and `loop` commands. PDF is largely based on PostScript but simplified to remove flow control features like these, while graphics commands such as `lineto` remain.

Often, the PostScript-like PDF code is generated from a source PostScript file. The graphics commands that are output by the PostScript code are collected and tokenized. Any files, graphics, or fonts to which the document refers also are collected. Then, everything is compressed to a single file. Therefore, the entire PostScript world (fonts, layout, measurements) remains intact.

As a document format, PDF has several advantages over PostScript:

- PDF contains tokenized and interpreted results of the PostScript source code, for direct correspondence between changes to items in the PDF page description and changes to the resulting page appearance.
- PDF (from version 1.4) supports graphic transparency; PostScript does not.
- PostScript is an interpreted programming language with an implicit global state, so instructions accompanying the description of one page can affect the appearance of any following page. Therefore, all preceding pages in a PostScript document must be processed to determine the correct appearance of a given page, whereas each page in a PDF document is unaffected by the others. As a result, PDF viewers allow the user to quickly jump to the final pages of a long document, whereas a PostScript viewer needs to process all pages sequentially before being able to display the destination page (unless the optional PostScript Document Structuring Conventions have been carefully complied with).

Technical overview

File structure

A PDF file is a 7-bit ASCII file, except for certain elements that may have binary content. A PDF file starts with a header containing the magic number and the version of the format such as `%PDF-1.7`. The format is a subset of a COS ("Carousel" Object Structure) format.[14] A COS tree file consists primarily of *objects*, of which there are eight types:[15]

- Boolean values, representing *true* or *false*
- Numbers
- Strings, enclosed within parentheses (`(...)`), may contain 8-bit characters.
- Names, starting with a forward slash (`/`)
- Arrays, ordered collections of objects enclosed within square brackets (`[...]`)
- Dictionaries, collections of objects indexed by Names enclosed within double pointy brackets (`<<...>>`)
- Streams, usually containing large amounts of data, which can be compressed and binary
- The null object

Furthermore, there may be comments, introduced with the percent sign (`%`). Comments may contain 8-bit characters.

Objects may be either *direct* (embedded in another object) or *indirect*. Indirect objects are numbered with an *object number* and a *generation number* and defined between the `obj` and `endobj` keywords. An index table, also called the cross-reference table and marked with the `xref` keyword, follows the main body and gives the byte offset of each indirect object from the start of the file.[16] This design allows for efficient random access to the objects in the file, and also allows for small changes to be made without rewriting the entire file (*incremental update*). Beginning with PDF version 1.5, indirect objects may also be located in special streams known as *object streams*. This technique reduces the size of files that have large numbers of small indirect objects and is especially useful for *Tagged PDF*.

At the end of a PDF file is a trailer introduced with the `trailer` keyword. It contains

- A dictionary
- An offset to the start of the cross-reference table (the table starting with the `xref` keyword)
- And the `%%EOF` end-of-file marker.

The dictionary contains

- A reference to the root object of the tree structure, also known as the *catalog*
- The count of indirect objects in the cross-reference table
- And other optional information.

There are two layouts to the PDF files: non-linear (not "optimized") and linear ("optimized"). Non-linear PDF files consume less disk space than their linear counterparts, though they are slower to access because portions of the data required to assemble pages of the document are scattered throughout the PDF file. Linear PDF files (also called "optimized" or "web optimized" PDF files) are constructed in a manner that enables them to be read in a Web browser plugin without waiting for the entire file to download, since they are written to disk in a linear (as in page order) fashion.[17] PDF files may be optimized using Adobe Acrobat software or QPDF.

Imaging model

The basic design of how graphics are represented in PDF is very similar to that of PostScript, except for the use of transparency, which was added in PDF 1.4.

PDF graphics use a device-independent Cartesian coordinate system to describe the surface of a page. A PDF page description can use a matrix to scale, rotate, or skew graphical elements. A key concept in PDF is that of the *graphics state*, which is a collection of graphical parameters that may be changed, saved, and restored by a *page description*. PDF has (as of version 1.6) 24 graphics state properties, of which some of the most important are:

- The *current transformation matrix* (CTM), which determines the coordinate system
- The *clipping path*
- The *color space*
- The *alpha constant*, which is a key component of transparency

Vector graphics

As in PostScript, vector graphics in PDF are constructed with *paths*. Paths are usually composed of lines and cubic Bézier curves, but can also be constructed from the outlines of text. Unlike PostScript, PDF does not allow a single path to mix text outlines with lines and curves. Paths can be stroked, filled, clipping. Strokes and fills can use any color set in the graphics state, including *patterns*.

PDF supports several types of patterns. The simplest is the *tiling pattern* in which a piece of artwork is specified to be drawn repeatedly. This may be a *colored tiling pattern*, with the colors specified in the pattern object, or an *uncolored tiling pattern*, which defers color specification to the time the pattern is drawn. Beginning with PDF 1.3 there is also a *shading pattern*, which draws continuously varying colors. There are seven types of shading pattern of which the simplest are the *axial shade* (Type 2) and *radial shade* (Type 3).

Raster images

Raster images in PDF (called *Image XObjects*) are represented by dictionaries with an associated stream. The dictionary describes properties of the image, and the stream contains the image data. (Less commonly, a raster image may be embedded directly in a page description as an *inline image*.) Images are typically *filtered* for compression purposes. Image filters supported in PDF include the general purpose filters

- **ASCII85Decode** a filter used to put the stream into 7-bit ASCII
- **ASCIIHexDecode** similar to ASCII85Decode but less compact
- **FlateDecode** a commonly used filter based on the deflate algorithm defined in RFC 1951 (deflate is also used in the gzip, PNG, and zip file formats among others); introduced in PDF 1.2; it can use one of two groups of predictor functions for more compact zlib/deflate compression: *Predictor 2* from the TIFF 6.0 specification and predictors (filters) from the PNG specification (RFC 2083)
- **LZWDecode** a filter based on LZW Compression; it can use one of two groups of predictor functions for more compact LZW compression: *Predictor 2* from the TIFF 6.0 specification and predictors (filters) from the PNG specification
- **RunLengthDecode** a simple compression method for streams with repetitive data using the run-length encoding algorithm and the image-specific filters
- **DCTDecode** a lossy filter based on the JPEG standard