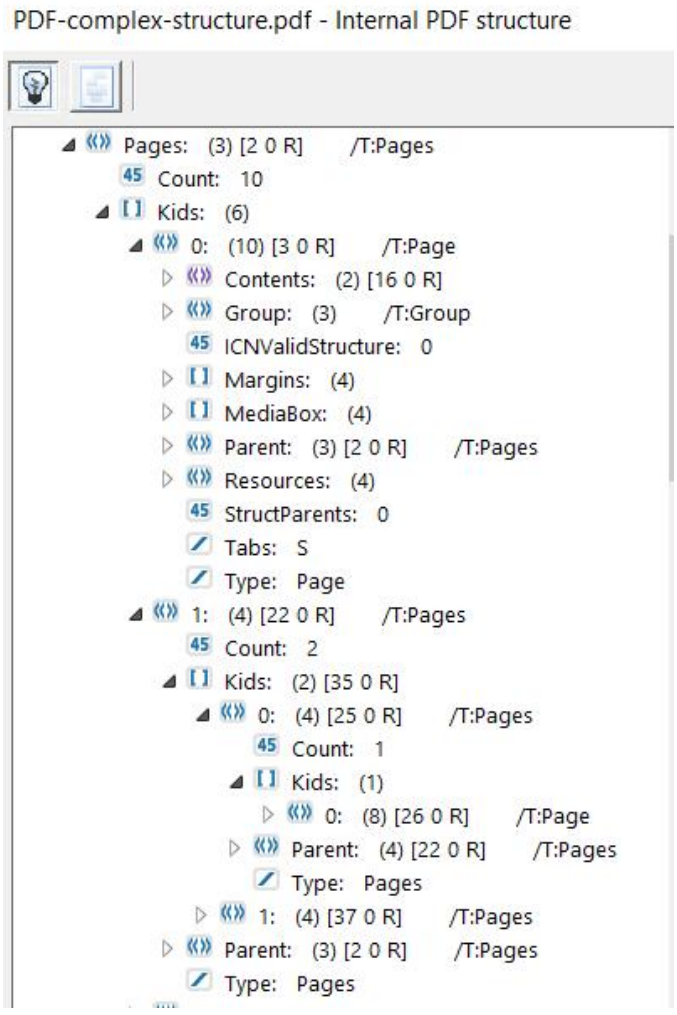
PDF document with a nested structure

This is an example of a document with a nested object structure, allowing the demonstration of the error "**Array value expected.**" when working with **FPDI - Free PDF Document Importer v2.6.2**.

Error example:

Fatal error: Uncaught exception 'setasign\Fpdi\PdfParser\Type \PdfTypeException' with message 'Array value expected.' in fpdi/src/PdfParser/ Type/PdfType.php:63

Stack trace:
#0 fpdi/src/PdfParser/Type/PdfArray.php(74): setasign\Fpdi\PdfParser\Type\PdfType::ensureType('setasign\ \Fpdi\\P...', Object(setasign\Fpdi\PdfParser\Type\PdfIndirectObjectReference), 'Array value exp...')
#1 fpdi/src/PdfReader/PdfReader.php(207): setasign\Fpdi\PdfParser\Type\PdfArray::ensure(Object(setasign \Fpdi\PdfParser\Type\PdfIndirectObjectReference))
#2 fpdi/src/PdfReader/PdfReader.php(222): setasign\Fpdi\PdfReader\PdfReader->setasign\Fpdi\PdfReader\ {closure}(Object(setasign\Fpdi\PdfParser\Type\PdfIndirectObjectReference), Object(setasign\Fpdi\PdfParser \Type\PdfNumeric))
#3 fpdi/src/PdfReader/PdfReader.php(243): setasign\Fpdi\PdfReader\PdfReader->setasign\Fpdi\PdfReader\ {closure}(Object(setasign\Fpdi\ in **fpdi/src/PdfParser/Type/PdfType.php** on line **63**



Example of PHP code causing an error

```
<?php

use setasign\Fpdi\Fpdi;

require_once('fpdf/fpdf.php');
require_once('fpdi/src/autoload.php');

class Pdf {

		function generatePreview($file)
		{
				$pdf = new FPDI();

				$pdf_handle = fopen($file, 'rb');
				$pageCount = $pdf->setSourceFile($pdf_handle);

				$max = min(5, $pageCount);

				for ($i=1; $i <= $max; $i++)
				{
						//  Add page to the document
						$templateID = $pdf->importPage($i);
						//$pdf->getTemplateSize($templateID);
						$pdf->addPage();
						$pdf->useTemplate($templateID);
				}

				$pdf_content = $pdf->Output('S');

				header("Content-type:application/pdf");
				echo $pdf_content;

				fclose($pdf_handle);

				return;
		}
}

require("libs/Pdf.php");

$pdf = new Pdf();

$pdf->generatePreview("files/PDF-complex-structure.pdf");
```

Portable Document Format

Adobe PDF icon
Filename extension **.pdf**
Internet media type
application/pdf,[1]
application/x-pdf
application/x-bzpdf
application/x-gzpdf
Type code **PDF [1] (including a single trailing space)**
Uniform Type Identifier (UTI) **com.adobe.pdf**
Magic number **%PDF**
Developed by **Adobe Inc. (1991–2008)**
ISO (2008–)
Initial release **June 15, 1993; 31 years ago**
Latest release **2.0**
Extended to **PDF/A, PDF/E, PDF/UA, PDF/VT, PDF/X**
Standard **ISO 32000-2**
Open format? **Yes**
Website **iso.org/standard/75839.html**

Portable Document Format (PDF), standardized as ISO 32000, is a file format developed by Adobe in 1992 to present documents, including text formatting and images, in a manner independent of application software, hardware, and operating systems.[2][3] 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 has its roots in “The Camelot Project” initiated by Adobe co-founder John Warnock in 1991.[4] PDF was standardized as ISO 32000 in 2008.[5] The last edition as ISO 32000-2:2020 was published in December 2020.

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), 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.

History

Main article: History of PDF
The development of PDF began in 1991 when John Warnock wrote a paper for a project then code-named Camelot, in which he proposed the creation of a simplified version of PostScript called Interchange PostScript (IPS).[6] Unlike traditional PostScript, which was tightly focused on rendering print jobs to output devices, IPS would be optimized for displaying pages to any screen and any platform.[6]

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 several other formats, including 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,[7][8] 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 necessary to make, use, sell, and distribute PDF-compliant implementations.[9]

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.[10] These proprietary technologies are not standardized, and their specification is published only on Adobe's website.[11][12][13] Many of them are not supported by popular third-party implementations of PDF.

ISO published version 2.0 of PDF, ISO 32000-2 in 2017, available for purchase, replacing the free specification provided by Adobe.[14] In December 2020, the second edition of PDF 2.0, ISO 32000-2:2020, was published, with clarifications, corrections, and critical updates to normative references[15] (ISO 32000-2 does not include any proprietary technologies as normative references).[16] In April 2023 the PDF Association made ISO 32000-2 available for download free of charge.[14]

Technical details

A PDF file is often a combination of vector graphics, text, and bitmap graphics. The basic types of content in a PDF are:

Typeset text stored as content streams (i.e., not encoded in plain text);
Vector graphics for illustrations and designs that consist of shapes and lines;
Raster graphics for photographs and other types of images; and
Other multimedia objects.
In later PDF revisions, a PDF document can also support links (inside document or web page), forms, JavaScript (initially available as a plugin for Acrobat 3.0), or any other types of embedded contents that can be handled using plug-ins.

PDF combines three technologies:

An equivalent subset of the PostScript page description programming language but in declarative form, 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 language

PostScript is a page description language run in an interpreter to generate an image.[6] It can handle graphics and has standard features of programming languages such as branching and looping.[6] PDF is a subset of PostScript, simplified to remove such control flow features, while graphics commands remain.[6]

PostScript was originally designed for a drastically different use case: transmission of one-way linear print jobs in which the PostScript interpreter would collect a series of commands until it encountered the showpage command, then execute all the commands to render a page as a raster image to a printing device.[17] PostScript was not intended for long-term storage and real-time interactive rendering of electronic documents to computer monitors, so there was no need to support anything other than consecutive rendering of pages.[17] If there was an error in the final printed output, the user would correct it at the application level and send a new print job in the form of an entirely new PostScript file. Thus, any given page in a PostScript file could be accurately rendered only as the cumulative result of executing all preceding commands to draw all previous pages—any of which could affect subsequent pages—plus the commands to draw that particular page, and there was no easy way to bypass that process to skip around to different pages.[17]

File format

A PDF file is organized using ASCII characters, except for certain elements that may have binary content. The file starts with a header containing a magic number (as a readable string) and the version of the format, for example %PDF-1.7. The format is a subset of a COS (”Carousel” Object Structure) format.[23] A COS tree file consists primarily of objects, of which there are nine types:[16]

Boolean values, representing true or false
Real numbers
Integers
Strings, enclosed within parentheses ((...)) or represented as hexadecimal within single angle brackets (<...>). Strings 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 angle brackets (<<...>>)
Streams, usually containing large amounts of optionally compressed binary data, preceded by a dictionary and enclosed between the stream and endstream keywords.
The null object
Comments using 8-bit characters prefixed with the percent sign (%) may be inserted.

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 if residing in the document root. Beginning with PDF version 1.5, indirect objects (except other streams) may also be located in special streams known as object streams (marked /Type /ObjStm). This technique enables non-stream objects to have standard stream filters applied to them, reduces the size of files that have large numbers of small indirect objects and is especially useful for Tagged PDF. Object streams do not support specifying an object’s generation number (other than 0).

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 2.0) 25 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
Black point compensation control (introduced in PDF 2.0)

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, fill then stroked, or used for 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 patterns of which the simplest are the axial shading (Type 2) and radial shading (Type 3).

Raster images

Raster images in PDF (called Image XObjects) are represented by dictionaries with an associated stream. The dictionary describes the properties of the image, and the stream contains the image data. (Less commonly, small raster images 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 following 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,
CCITTFaxDecode, a lossless bi-level (black/white) filter based on the Group 3 or Group 4 CCITT (ITU-T) fax compression standard defined in ITU-T T.4 and T.6,
JBIG2Decode, a lossy or lossless bi-level (black/white) filter based on the JBIG2 standard, introduced in PDF 1.4, and
JPXDecode, a lossy or lossless filter based on the JPEG 2000 standard, introduced in PDF 1.5.

Normally all image content in a PDF is embedded in the file. But PDF allows image data to be stored in external files by the use of external streams or Alternate Images. Standardized subsets of PDF, including PDF/A and PDF/X, prohibit these features.

Additional features

Logical structure and accessibility
See also: PDF/A-1 and PDF/UA
A tagged PDF (see clause 14.8 in ISO 32000) includes document structure and semantics information to enable reliable text extraction and accessibility.[30] Technically speaking, tagged PDF is a stylized use of the format that builds on the logical structure framework introduced in PDF 1.3. Tagged PDF defines a set of standard structure types and attributes that allow page content (text, graphics, and images) to be extracted and reused for other purposes.[31]

Tagged PDF is not required in situations where a PDF file is intended only for print. Since the feature is optional, and since the rules for tagged PDF were relatively vague in ISO 32000-1, support for tagged PDF among consuming devices, including assistive technology (AT), is uneven as of 2021.[32] ISO 32000-2, however, includes an improved discussion of tagged PDF which is anticipated to facilitate further adoption.

An ISO-standardized subset of PDF specifically targeted at accessibility, PDF/UA, was first published in 2012.

Optional Content Groups (layers)
With the introduction of PDF version 1.5 (2003) came the concept of Layers. Layers, more formally known as Optional Content Groups (OCGs), refer to sections of content in a PDF document that can be selectively viewed or hidden by document authors or viewers. This capability is useful in CAD drawings, layered artwork, maps, multi-language documents, etc.

Basically, it consists of an Optional Content Properties Dictionary added to the document root. This dictionary contains an array of Optional Content Groups (OCGs), each describing a set of information and each of which may be individually displayed or suppressed, plus a set of Optional Content Configuration Dictionaries, which give the status (Displayed or Suppressed) of the given OCGs.