PDF Reference

sixth edition

Adobe® Portable Document Format

Version 1.7

November 2006

Adobe Systems Incorporated

CHAPTER 2

Overview

PDF is a file format for representing documents in a manner independent of the application software, hardware, and operating system used to create them and of the output device on which they are to be displayed or printed. A *PDF document* consists of a collection of *objects* that together describe the appearance of one or more *pages*, possibly accompanied by additional interactive elements and higher-level application data. A *PDF file* contains the objects making up a PDF document along with associated structural information, all represented as a single self-contained sequence of bytes.

A document's pages (and other visual elements) can contain any combination of text, graphics, and images. A page's appearance is described by a PDF *content stream*, which contains a sequence of *graphics objects* to be painted on the page. This appearance is fully specified; all layout and formatting decisions have already been made by the application generating the content stream.

In addition to describing the static appearance of pages, a PDF document can contain interactive elements that are possible only in an electronic representation. PDF supports *annotations* of many kinds for such things as text notes, hypertext links, markup, file attachments, sounds, and movies. A document can define its own user interface; keyboard and mouse input can trigger *actions* that are specified by PDF objects. The document can contain *interactive form* fields to be filled in by the user, and can export the values of these fields to or import them from other applications.

Finally, a PDF document can contain higher-level information that is useful for interchange of content among applications. In addition to specifying appearance, a document's content can include identification and logical structure information

that allows it to be searched, edited, or extracted for reuse elsewhere. PDF is particularly well suited for representing a document as it moves through successive stages of a prepress production workflow.

2.1 Imaging Model

At the heart of PDF is its ability to describe the appearance of sophisticated graphics and typography. This ability is achieved through the use of the *Adobe imaging model*, the same high-level, device-independent representation used in the PostScript page description language.

Although application programs could theoretically describe any page as a full-resolution pixel array, the resulting file would be bulky, device-dependent, and impractical for high-resolution devices. A high-level imaging model enables applications to describe the appearance of pages containing text, graphical shapes, and sampled images in terms of abstract graphical elements rather than directly in terms of device pixels. Such a description is economical and device-independent, and can be used to produce high-quality output on a broad range of printers, displays, and other output devices.

2.1.1 Page Description Languages

Among its other roles, PDF serves as a *page description language*, a language for describing the graphical appearance of pages with respect to an imaging model. An application program produces output through a two-stage process:

1. The application generates a device-independent description of the desired output in the page description language.
2. A program controlling a specific output device interprets the description and *renders* it on that device.

The two stages may be executed in different places and at different times. The page description language serves as an interchange standard for the compact, device-independent transmission and storage of printable or displayable documents.

G.2 Simple Text String Example

Example G.2 is the classic "Hello World" example built from the preceding example. It shows a single line of text consisting of the string Hello World, illustrating the use of fonts and several text-related PDF operators. The string is displayed in 24-point Helvetica. Because Helvetica is one of the standard 14 fonts, no font descriptor is needed.

Table G.2 lists the objects in this example.

TABLE G.2 Objects in simple text string example

OBJECT NUMBER	OBJECT TYPE
1	**Catalog** (document catalog)
2	**Outlines** (outline dictionary)
3	**Pages** (page tree node)
4	**Page** (page object)
5	Content stream
6	Procedure set array
7	**Font** (Type 1 font)

Example G.2

```
%PDF–1.4
1  0  obj
   <<  /Type  /Catalog
       /Outlines  2 0 R
       /Pages  3 0 R
   >>
endobj

2  0  obj
   <<  /Type  /Outlines
       /Count  0
   >>
endobj
```

```
3 0 obj
   << /Type /Pages
      /Kids [4 0 R]
      /Count 1
   >>
endobj

4 0 obj
   << /Type /Page
      /Parent 3 0 R
      /MediaBox [0 0 612 792]
      /Contents 5 0 R
      /Resources << /ProcSet 6 0 R
                    /Font << /F1 7 0 R >>
                 >>
   >>
endobj

5 0 obj
   << /Length 73 >>
stream
   BT
      /F1 24 Tf
      100 100 Td
      (Hello World) Tj
   ET
endstream
endobj

6 0 obj
   [/PDF /Text]
endobj

7 0 obj
   << /Type /Font
      /Subtype /Type1
      /Name /F1
      /BaseFont /Helvetica
      /Encoding /MacRomanEncoding
   >>
endobj
```

```
xref
0 8
0000000000 65535 f
0000000009 00000 n
0000000074 00000 n
0000000120 00000 n
0000000179 00000 n
0000000364 00000 n
0000000466 00000 n
0000000496 00000 n

trailer
   << /Size 8
      /Root 1 0 R
   >>
startxref
625
%%EOF
```

G.3 Simple Graphics Example

Example G.3 draws a thin black line segment, a thick black dashed line segment, a filled and stroked rectangle, and a filled and stroked cubic Bézier curve. Table G.3 lists the objects in this example, and Figure G.1 shows the resulting output. (Each shape has a red border, and the rectangle is filled with light blue.)

TABLE G.3 Objects in simple graphics example

OBJECT NUMBER	OBJECT TYPE
1	**Catalog** (document catalog)
2	**Outlines** (outline dictionary)
3	**Pages** (page tree node)
4	**Page** (page object)
5	Content stream
6	Procedure set array