Dr. Edgar Huckert
63773 Goldbach, Germany
E-Mail: Huckert@compuserve.com
6-97

PDF Essentials

PDF is an acronym for "Portable Document Format". The format has been designed by Adobe for the exchange of documents on various platforms. Currently the format is used under Windows, MAC and various UNIX platforms.

This is essentially a **final** format - in contrast to SGML based formats which are aimed to be **revisable.** "Final" means: it is not primarily aimed at being edited - although this is possible. The most visible difference compared to other popular formats is the complete absence of information about the logical structure of a document: there are no information in PDF about chapters, headers, footers, tables of content etc. The most important structural concept is the **page**: PDF is like its close parent Postscript a page oriented format.

PDF started in version 1.0 as a 7-bit encoded format. In later versions the size aspect became more important so that the 7-bit restriction was dropped. LZW compression for the text portions and JPEG or CCITT compressions without additional 7-bit encoding à la *uuencode* are now common in PDF documents. The other major addition to the set of PDF features was the enhancement of the link concept. You can now define links within a document, links to other documents and links to external applications. The link concept appears in PDF as a generalized annotation concept.

The PDF specification has been published by Adobe. There is no major risk that PDF documents can't be understood in the future as the internal structure of PDF is known. This is its major advantage over formats like DOC used by Microsoft WORD whose internals are not known.

The first Adobe PDF spec (version 1.0) was published as the normal book. The last PDF specification I have is the version 1.1. At the time of this writing a newer version 1.2 should be available. The specs can be found on the Adobe ftp home. They can also be downloaded from the Adobe web home.

The Structure of PDF Documents

PDF documents consist mainly of four parts:

a very short header

the document body

the cross reference

the trailer

The **header** identifies a document as a PDF document via a string "%PDF-1.1". This is like a magic number in UNIX files.

The **document body** contains the information about the objects on the pages. This is a collection of objects organized as a tree describing the page structure, the pages and the elements (text, graphics etc.) on the pages. The PDF objects are the leafs of the PDF tree. Each object has three essential components: it has a number, it has a fixed position in the PDF file (an offset) and it has a content.

The **cross reference** allows a PDF parser (like the Acrobat reader) to quickly access the objects. It starts with the keyword *xref* followed by the number of objects. An entry in the cross reference is mainly an offset in the document file and a flag indicating whether this object is free (can be reused) or occupied. For advanced applications multiple cross references can be kept in a PDF file thus allowing to maintain a *chain of versions* for a document. Note that the cross reference doesn't allow much syntactic freedom: it must be written and read like a fixed header in formats like PCX. This is the start of a typical cross reference:

```
xref
0 20
0000000000 65535 f
0000000009 00000 n
0000000116 00000 n
.......
```

The **trailer** is a quick entry into the most essential document parts: it contains a pointer to the start of the cross reference and the object numbers for the most essential objects: the **root object** (this is the start of the tree of pages) and the **info object** containing some important meta information. This is a sample trailer:

```
trailer
<<
/Size 20
/Root 19 0 R
/Info 18 0 R
>>
startxref
2354
%%EOF
```

Whereas the PDF document body must be parsed by a program (the token must be identified and interpreted) the header, cross reference and the trailer must be handled like data structures.

The body of a PDF document

The body of a PDF document is essentially a tree of objects starting with the root object mentioned before. The root object has a reference to the */Pages* object; this is the start of the page

tree. Remember that PDF is a page oriented format - you will find no logical layout information in the documents. If you look into the */Pages* object you will see an array of type *„/Kids“:* this array contains the object numbers of the */Page* objects, i.e. of the document pages.

Each page starts with a */Page* object. Such a */Page* object has a set of page-wide attributes and a set of content attributes referenced by the */Contents* array. Here is a typical */Page* object:

```
17 0 obj
<<
/Type /Page
/Parent 11 0 R
/MediaBox [0 0 600 810]
/Resources
<<
/Font << /F1 1 0 R /F2 2 0 R >>
/ProcSet 10 0 R
>>
/Contents [16 0 R 15 0 R ]
>>
endobj
```

The array following the */MediaBox* keyword defines the size of the page. The */Resources* dictionary (dictionaries appear between „<<„ and „>>„) names to fonts to be used on the page. If you follow the reference to the */ProcSet* object you will see whether this page uses graphics oder video or audio components. Note that the page object also contains a reference to the parent object, i.e. to the */Pages* object.

Let’s now look at the operators used to draw text and simple line graphics. The operators used in PDF have much in common with the operators used in Postscript. The following extract from a PDF file is an object that draws an header on the page. It displays a text line followed by a horizontal line used as a separator:

```
6 0 obj
<< /Length 88 >>
stream
BT
/F1 11 Tf
1 0 0 1 25 798 Tm (kopfzeile         - 1 -) Tj
2 w
12 792 m
550 792 l S
ET
endstream
endobj
```

The important operators are her:

Tj	shows the text (enclosed in „(„ and „)“
Tm	defines the text matrix (the coordinates)
w	defines the line width
m	moves the cursor

A PDF sample document

Note: if you plan to write a simple PDF document with your favorite editor be careful with the offsets in the cross reference. You must know whether you favorite operating system expands linefeeds to CR + LF or not! If ever you transfer a PDF document from one operating system to an other operating system use the BINARY option with *ftp*.

This sample PDF document contains two pages. Each page only shows a single line of text. A header line appears at the top of each page. If you try to understand the structure of the PDF file remember that PDF documents form a tree and that you should start from the end of the file: you have to locate the root object (here object 12) and then the /Pages object (here object 4). From there on you find the two /Page objects mentioned as /Kids: the objects 7 and 10. The rest should be easy.

```
%PDF-1.1
1 0 obj
<<
/Type /Font
/Subtype /Type1
/Name /F1
/BaseFont /Helvetica
/Encoding /WinAnsiEncoding
>>
endobj
2 0 obj
<<
/Type /Font
/Subtype /Type1
/Name /F2
/BaseFont /Helvetica-Bold
/Encoding /WinAnsiEncoding
>>
endobj
3 0 obj
[/PDF /Text /ImageB]
endobj
5 0 obj
<< /Length 81 >>
stream
BT
/F1 13 Tf
1 0 0 1 25 762 Tm (Das ist nur ein Beispielstext auf Seite 1) Tj
ET
endstream
endobj
6 0 obj
<< /Length 88 >>
stream
BT
/F1 11 Tf
1 0 0 1 25 798 Tm (kopfzeile         - 1 -) Tj
```

```
2 w
12 792 m
550 792 l S
ET
endstream
endobj
7 0 obj
<<
/Type /Page
/Parent 4 0 R
/MediaBox [0 0 600 810]
/Rotate 0
/Resources
<<
/Font <<
/F1 1 0 R /F2 2 0 R  >>
/ProcSet 3 0 R
>>
/Contents [6 0 R 5 0 R ]
>>
endobj
8 0 obj
<< /Length 90 >>
stream
BT
/F1 13 Tf
1 0 0 1 25 774 Tm () Tj
/F1 13 Tf
1 0 0 1 25 762 Tm (Text auf Seite 2) Tj
ET
endstream
endobj
9 0 obj
<< /Length 88 >>
stream
BT
/F1 11 Tf
1 0 0 1 25 798 Tm (kopfzeile         - 2 -) Tj
2 w
12 792 m
550 792 l S
ET
endstream
endobj
10 0 obj
<<
/Type /Page
/Parent 4 0 R
/MediaBox [0 0 600 810]
/Rotate 0
/Resources
<<
/Font <<
/F1 1 0 R /F2 2 0 R  >>
/ProcSet 3 0 R
>>
/Contents [9 0 R 8 0 R ]
>>
endobj
11 0 obj
<<
/Subject (kopfzeile)
```

```
/Title (mktxtpdf demo)
/Creator (Program mktxtpdf)
/Author (Dr.E.Huckert, Goldbach (Germany))
/CreationDate (21.06.1997)
>>
endobj
4 0 obj
<<
/Type /Pages
/Count 2
/Kids [7 0 R 10 0 R ]
>>
endobj
12 0 obj
<<
/Type /Catalog
/Pages 4 0 R
>>
endobj
xref
0 13
0000000000 65535 f
0000000009 00000 n
0000000116 00000 n
0000000228 00000 n
0000001314 00000 n
0000000264 00000 n
0000000394 00000 n
0000000531 00000 n
0000000702 00000 n
0000000841 00000 n
0000000978 00000 n
0000001150 00000 n
0000001379 00000 n
trailer
<<
/Size 13
/Root 12 0 R
/Info 11 0 R
>>
startxref
1429
%%EOF
```