PDF Specifications

Portable Document Format, PDF, is a file format similar to Postscript. It enables users to exchange and view electronic documents independent of the platform on which they were created. This allows any user with Adobe Acrobat Reader to view a PDF document.

The PDF implementation of an Employee Directory was created using basic structures of the PDF format. The following is an overview of the structures used to create the document.

Basic Document Structure

The PDF document is defined in segments called objects. Each object has a number assigned to it so that it can be referred to by other objects. The objects do not have to appear in chronological order in the document, but is much easier to follow when they are.

The first part of a PDF document is the Catalog. The Catalog is the root object and contains references to the pages and outline contained in the document. The reference is a pointer to another object that defines the pages and outlines. Each page can be represented by with it's own object. The Catalog references an object that contains the number of pages and the object number of each page. The reference to the outline contains the number of outlines and their object numbers. If no outline is used, the number of outlines is set to zero.

The Page object defines how the page will appear. The fonts that will be used, which are defined in their own object, are referenced, as well as the parent object, the ProcSet, MediaBox, and the object that contains the actual information. These are explained in more detail later. The information that the page contains is defined in a stream. The length of this stream in characters must be know, and is referenced in another object. If the length of the stream is known before the PDF document is written, the length of the stream can be written into the document without the need of another object.

After all of the objects have been written, the xref is made. This is a listing of the objects and position that the object starts in the file in bytes. It is possible for PDF readers to jump to a particular page in a document. The xref tells the reader where in the file to go to for the page without having to read through or download the whole document. After the xref is the trailer. This tells the reader the number of objects (including one free object) are in the document and the object number of the root object. Finally, the total length of the file up to the xref is written, followed by an end of file marker.

Syntax

The first line of the PDF document is the version of the PDF format being used. For simple documents only containing text, version 1.0 can be used.

The first line of an object starts with the object number, followed by the generation number, and finally the keyword obj. The generation number is generally set to zero, and obj lets the reader know that the following information is an object. The content of the object is contained within opening << and closing >>. The object is closed by the endobj keyword.

The first object is the Catalog. The object is defined as a Catalog using the Type keyword. The Pages keyword is used to reference the object that contains information about the page objects. If an outline is used, the Outlines keyword references the object that contains the outline objects. The following is part of a PDF file showing the version and Catalog object.

```
%PDF-1.0
1 0 obj
<<
/Type /Catalog
/Pages 3 0 R
/Outlines 2 0 R
>>
endobj
2 0 obj
<<
/Type /Outlines
/Count 0
>>
endobj
3 0 obj
<<
/Type /Pages
/Count 1
/Kids [4 0 R]
>>
endobj
```

The page information is contained in object three and the outline information is contained in object two. Object two is of type outline and /Count 0, shows that it has no information defined for the outline. Object three is of type pages, so it defines the page objects for the document. The number of pages is defined using the count keyword. /Kids references the objects that contain the page format information. If there were two pages, /Kids [4 0 R 25 0 R] would be used where object 25 is an object containing the page format information.

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

Object four is of type page and defines the page format. It references it's parent object, which is the object that calls it, and the object that contains the page information. The /Resources tells the main format of the page to be used. The objects that define the fonts and ProcSets are defined inside the opening << and closing >>. The MediaBox defines the size of the page in points in a rectangular coordinate system. The lower left corner is (0,0) and in this example the upper right corner is (612,792). One inch is equal to 72 points. Positioning of characters on the page is done using this coordinate system. /Contents references the object, 5, that contains the page information.

```
5 0 obj
<< /Length 44 >>
stream
BT
/F1 24 Tf
100 100 Td (Hello World) Tj
ET
endstream
endobj
```

Object five is the main object that defines what appears on the page. Since the information is contained in the stream, the length of the stream has to be known. In this example, the length if the stream is known to be 44 bytes. This can be verified by counting the characters, including an end of line character, starting with the BT and ending with the ET. In general, if the length of the stream is not know, an object that appears after the page object must be referenced that contains the length of the stream. This is accomplished by changing << /Length 44>> to /Length X 0 R, where X is the number of the object that contains the length of the stream. This object is very simple:

```
X 0 obj
44
endobj
```

After the length declaration, the information stream begins. The keywords steam and BT start the stream. Likewise, ET and endstream end it. Before text can be written, it has to have a font referenced. The name of the font and the point size for the text is declared, followed by the keyword Tf. Tf sets the font name and size.

This way of defining a page makes it necessary to have three objects per page. These objects are of types Pages, Page, and the length object.

The next line contains the text and it's placement on the page. If this is the first text to be written to the page, the current x and y position of the page are both zero. After text has been written, the current x and y positions take on the value that x and y were increased or decreased by. The Td keyword is used to move x and y positions. The difference in the x position, the difference in the y position, preceeds Td to move the text position. So to move to (50,50), before any other text has been placed, "50 50 Td" would be used. Now to move to (100, 25), "50 -25 Td" is used. The length of the text does not change the x and y positions.

The text to be written to the page follows the Td and is contained within parenthesis. The Tj keyword tells the reader to show the text. If the text has hanging parenthesis, this will cause the page containing the hanging paren to not load.

```
6 0 obj
[/PDF /Text]
endobj
7 0 obj
<<
/Type /Font
/Subtype /Type1
/Name /F1
/BaseFont /Helvetica
>>
endobj
```

Object six defines the ProcSet to be used. For a simple document that only contains text, /PDF and /Text are the only ProcSet's needed. There are other ProcSet's that need to be used of images are to be included.

Object seven is of type font which defines the font to be used. The Sybtype keyword defines the type of font to be used. Type1 fonts are special-purpose PostScript language programs used for defining fonts. The name of the font, as it is to be used in the PDF document, is defined using the name keyword. The name of the font has the format /FX, where X is an integer. Multiple fonts can be defined in this way. There are 14 Type1 fonts that are guaranteed to be present in the reader. These fonts do not need any attributes defined for them. The BaseFont can be one of these 14 fonts or a special font can be defined.

```
xref
0 8
0000000000 65535 f
0000000009 00000 n
0000000074 00000 n
0000000120 00000 n
0000000179 00000 n
0000000322 00000 n
0000000415 00000 n
0000000445 00000 n
```

After all of the objects have been written, the xref is made. Each object in a PDF file is referenced in the xref table at the end of the file. This table begins with the xref keyword. The next line begins with a number representing the subsection of the xref table. For simple PDF files, this number is usually 0. The following number is the number of objects used in the file, plus one for the next free object to be used. This free object is defined on the following line. The remaining lines are objects that are in use. The first set of numbers is the number of bytes into the file where the object starts. The second set of numbers is the generation number of the object. This number is originally set to 00000, (unless it is a free object, where it is set to 65535). Each time that the object is updated or changed, the generation number is increased by 1, until the maximum number, 65535, is reached. The generation number is followed by either an f, for free object, or an n, for a used object. There are seven objects in use, plus the one free object for a total of 8, as seen on the second line. The first object in use, begins at 9 bytes into the file. It has not been updated or changed because the generation number is still 00000. The last object begins 445 bytes into the file, and also has not been modified.

```
trailer
<<
/Size 8
/Root 1 0 R
>>
startxref
553
%%EOF
```

After the xref is the trailer. This reports the total number of objects, including the free object. It also tells the root document, which is the Catalog, object number one. The trailer is contained in opening << and closing >>. When the reader opens the the PDF document, it starts at the end of the file and reads the trailer and the xref first. This is why startxref comes after the xref has been defined. Reading the end of the file first allows the reader to know how far into the document each object starts so that individual pages can be accessed without having to read or download the entire document. After startxref, the number of bytes into the file where the keyword xref begins, is reported. Finally, %%EOF marks the end of the document.

The 14 Type1 BaseFonts Guaranteed to be in Adobe Acrobat Reader

```
Courier
Courier-Bold
Courier-Oblique
Courier-BoldOblique
Symbol
Helvetica
Helvetica-Bold
Helvetica-Oblique
Helvetica-BoldOblique
Times-Roman
Times-Bold
Times-Italic
Times-BoldItalic
ZapfDingbats
```

APPENDIX B

Summary of Page Marking Operators

Following is a list of all page marking operators used in PDF files, arranged alphabetically. For each operator, a brief description is given, along with a reference to the page in this document where the operator is discussed in detail. Words shown in boldface in the summary column are PostScript language operators.

Table B.1 *PDF page marking operators*

Operator	*Summary*	*Page*	
b	**closepath**, **fill**, and **stroke** path	227	
B	**fill** and **stroke** path	227	
b*	**closepath**, **eofill**, and **stroke** path	227	
B*	**eofill** and **stroke** path	227	
BDC	begin marked content, with a dictionary	240	**PDF 1.2**
BI	begin image	238	
BMC	begin marked content	240	**PDF 1.2**
BT	begin text object	233	
BX	begin section allowing undefined operators	239	**PDF 1.1**
c	**curveto**	223	
cm	**concat**. Concatenates the matrix to the current transformation matrix.	213	
cs	**setcolorspace** for fill	220	**PDF 1.1**
CS	**setcolorspace** for stroke	221	**PDF 1.1**
d	**setdash**	215	
d0	**setcharwidth** for Type 3 font	239	
d1	**setcachedevice** for Type 3 font	239	
Do	execute the named XObject	236	

Table B.1 *PDF page marking operators*

Operator	*Summary*	*Page*	
DP	mark a place in the content stream, with a dictionary	241	**PDF 1.2**
EI	end image	238	
EMC	end marked content	240	**PDF 1.2**
ET	end text object	233	
EX	end section that allows undefined operators	239	**PDF 1.1**
f	**fill** path	226	
F	**fill** path	226	
f*	**eofill** path	226	
g	**setgray** (fill)	219	
G	**setgray** (stroke)	220	
gs	set parameters in the extended graphics state	217	**PDF 1.2**
h	**closepath**	225	
i	**setflat**	213	
ID	begin image data	238	
j	**setlinejoin**	216	
J	**setlinecap**	214	
k	**setcmykcolor** (fill)	220	
K	**setcmykcolor** (stroke)	220	
l	**lineto**	223	
m	**moveto**	223	
M	**setmiterlimit**	217	
MP	mark a place in the content stream	241	**PDF 1.2**
n	end path without **fill** or **stroke**	226	
q	save graphics state	213	
Q	restore graphics state	213	
re	rectangle	224	
rg	**setrgbcolor** (fill)	220	
RG	**setrgbcolor** (stroke)	220	

Table B.1 *PDF page marking operators*

Operator	*Summary*	*Page*	
s	**closepath** and **stroke** path	226	
S	**stroke** path	226	
sc	**setcolor** (fill)	221	**PDF 1.1**
SC	**setcolor** (stroke)	221	**PDF 1.1**
scn	**setcolor** (fill, in pattern and separation color spaces)	221	**PDF 1.2**
SCN	**setcolor** (stroke, in pattern and separation color spaces)	221	**PDF 1.2**
Tc	set character spacing	229	
Td	move text current point	233	
TD	move text current point and set leading	233	
Tf	set font name and size	230	
Tj	show text	234	
TJ	show text, allowing individual character positioning	235	
TL	set leading	230	
Tm	set text matrix	233	
Tr	set text rendering mode	232	
Ts	set super/subscripting text rise	232	
Tw	set word spacing	229	
Tz	set horizontal scaling	230	
T*	move to start of next line	233	
v	**curveto**	223	
w	**setlinewidth**	216	
W	**clip**	227	
W*	**eoclip**	227	
y	**curveto**	224	
'	move to next line and show text	234	
"	move to next line and show text	235	