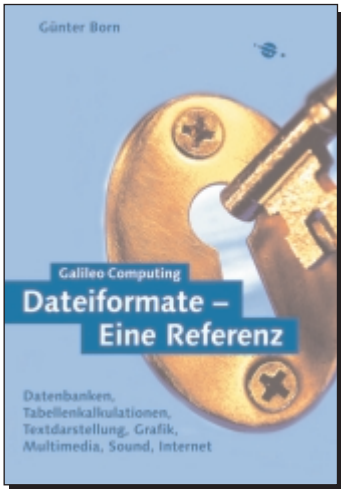
Dieses Kapitel stammt aus dem Buch

›Dateiformate – Eine Referenz‹

von Günter Born.

www.borncity.de



ISBN 3-934358-83-7
119,90 DM

Informationen zum Buch
mit Bestellmöglichkeit

www.galileocomputing.de

Galileo Computing

49 Das Portable Document Format (PDF)

Das von Adobe definierte Format PDF (nachfolgend auch als PDF-Format bezeichnet) erlaubt die Ausgabe und Anzeige von Dokumenten in einem anwendungsübergreifenden Format. Dateien im PDF-Format können Texte und Grafiken enthalten und lassen sich mit den von Adobe angebotenen Acrobat Readern unter Windows und Macintosh anzeigen. Die PDF-Version 1.0 wird auch unter Unix unterstützt.

Eine PDF-Datei besteht aus einem Header, einem Body (mit dem Dokumentinhalt), einer Querverweisliste (für Verweise) und dem Trailer. Diese Teile enthalten Befehle, die sich stark an PostScript anlehnen. Alle Informationen innerhalb einer PDF-Datei werden im ASCII-Format hinterlegt, und eine Zeile darf nicht mehr als 255 Zeichen umfassen. Binäre Daten sind als ASCII-Zeichen zu kodieren (Hexzeichen). Nachfolgend sehen Sie einen Ausschnitt aus einer PDF-Datei:

```
%PDF-1.1
%ãäÏÓ
1 0 obj
[
/CalRGB << /WhitePoint [ 0.9505 1 1.089 ] /Gamma [ 1.8 1.8 1.8 ] /Matrix [ 0.4497
0.2446 0.0252 0.3163 0.672 0.1412 0.1845 0.0833 0.9227 ] >>

]
endobj
2 0 obj
<<
/Type /Page
/Parent 10 0 R
/Resources 4 0 R
/Contents 3 0 R
/CropBox [ 57 379 517 754 ]
/Annots 121 0 R
>>
endobj
3 0 obj
<< /Length 2656 /Filter /LZWDecode >>
stream
<Streamdaten.......>
```

```
endstream
endobj
4 0 obj
<<
/ProcSet [ /PDF /Text ]
/ColorSpace << /DefaultRGB 1 0 R >>
/Font << /F2 5 0 R /F4 6 0 R /F6 7 0 R /F8 8 0 R /F10 9 0 R >>
>>
endobj
5 0 obj
<<
/Type /Font
/Subtype /Type1
/Name /F2
/FirstChar 32
/LastChar 255
/Widths [ ....
......
trailer
<<
/Size 172
/Info 101 0 R
/Root 100 0 R
/ID[<b64328e1c56ce225183bac5f0dff91bd><b64328e1c56ce225183bac5
f0dff91bd>]>>
startxref
186264
%%EOF
```

Abbildung 49.1 Auszug aus einer PDF-Datei

Hinweise zu PDF-Befehlen

PDF-Befehle stellen Objekte dar, die einem reduzierten PostScript-Befehlsumfang entsprechen. So können in PostScript-Dateien Verzweigungen, Schleifen, Prozeduren etc. genutzt werden, PDF-Dokumente bieten diese Möglichkeiten nicht, sondern besitzen einen linearen Programmablauf. Alle Befehle und Namen sind case-sensitiv, d.h. Groß-/Kleinschreibung wird unterschieden. In PDF-Dateien kommen Werte (als Objekte interpretiert) mit den Basistypen *boolean, number, string, name, array, dictionary* und *stream* vor. Ein *boolean*-Objekt kann die Werte *true* oder *false* annehmen.

Zahlen werden entweder als Integer oder als Real angegeben. Integerwerte sind Ganzzahlen, während Realwerte Dezimalzahlen darstellen. Bei Dezimalzahlen wird kein Exponentialformat (1.0E3) unterstützt. Zeichenketten (Strings) werden in runde Klammern gestellt. Die folgende Angabe enthält eine Zeichenkette:

```
(Dies ist ein Text)
(Ein Text mit \n
Zeilenumbruch)
```

Das Zeichen \ besitzt dabei eine besondere Funktion, es leitet eine sogenannte Escape-Sequenz ein. Das Folgezeichen besitzt dabei eine besondere Bedeutung:

```
\n    linefeed
\r    carriage return
\t    horizontaler Tabulator
\b    backspace
\f    formfeed
\\    backslash
\(    linke Klammer
\)    rechte Klammer
\ddd  oktale Zahl mit dem Zeichencode
```

Da die PDF-Dateien nur druckbare ASCII-Zeichen beinhalten, müssen alle nicht mit diesem Zeichensatz kodierbaren Zeichen als *\ddd* angegeben werden. Zusätzlich können Zeichenketten auch wie in PostScript über eine Sequenz an Hexadezimalzahlen in der Form *<40414b>* angegeben werden. Beachten Sie aber, daß Hexadezimalziffern in kleinen Buchstaben (a, b, c, d, e und f) anzugeben sind.

Namen werden als Zeichenketten in der Form /name angegeben. Der Name muß mit dem Slash-Zeichen gefolgt von einem Buchstaben beginnen. An den ersten Buchstaben können sich beliebige Zeichen mit Ausnahme *linefeed, carriage return, %, (,), <, >, [,], {* und *}* anschließen.

Zusätzlich können PDF-Objekte zu Sequenzen zusammengefaßt und in Feldern (Arrays) gespeichert werden. Arrays werden dabei mit eckigen Klammern *[0 1 /test]* zusammengefaßt. Bei Dictionaries handelt es sich um Tabellen, in denen Objektpaare (jeweils ein Schlüssel *key* und der eigentliche Wert *value*) zugeordnet werden. Ein Dictionary wird durch spitze Klammern << ... >> dargestellt (z. B. *<</Type /MeinBeispiel /name (Born) /vorname (Klaus)>>*). Der erste Eintrag im Dictionary ist */Type* gefolgt vom Namen des Objekts (hier */MeinBeispiel*). Dann folgen die Schlüssel-Werte-Paare. Dictionary-Objekte sind der Hauptbestandteil von PDF-Dateien, da Seiten oder Schriftarten (Fonts) über solche Objekte abgebildet werden. Mit */Type /Font* wird eine Schriftart referenziert. Dann tritt das Objekt */SubType* auf, welches die Schrifttechnologie (*/Type1, MMType1, Type3* oder */TrueType*) angibt.

Streams sind Zeichenfolgen, die durch eine Anwendung sequentiell gelesen werden können. Streams eignen sich daher zur Speicherung großer Datenmengen (wie Bilddaten oder Seitenbeschreibungen). Ein Stream besteht aus einem Dictionary-Objekt, welches die Zeichenfolge beschreibt, gefolgt vom Schlüsselwort *stream*, gefolgt von einer oder mehreren Zeilen mit den Zeichen (dem Text) und abgeschlossen vom Schlüsselwort *endstream*:

```
<< dictionary object>> stream
( Textzeilen )
endstream
```

Das Dictionary-Objekt enthält verschiedene Schlüsselwörter. */Length* ist erforderlich und gibt die Länge (Zeichenzahl) des auf das Schlüsselwort *stream* folgenden Strings bis zu *endstream* an. Die Daten können komprimiert vorliegen. Mit dem optionalen Wort */Filter* kann ein Filter zum Entschlüsseln des Datenstroms angegeben werden. Die betreffenden Einträge werden als Feld angegeben (z. B. */Filter [/ASCII85Decode /LZWDecode]*). Hierbei werden die Standard-PostScript Level 2-Dekodierungsfilter (siehe vorheriges Kapitel) unterstützt (ASCIIHexDecode = Binärdaten werden als Hexadezimalzahlenfolgen dargestellt; ASCII85Decode = Binärdaten werden als ASCII bas-85 interpretiert, LZWDecode = hier sind Texte oder Binärdaten mit einem LZW-Algorithmus komprimiert, RunLengthDecode = entpacken der Daten mit einem Run Length-Algorithmus, CCITTFaxDecode = Binärdaten werden nach dem CCITT-Fax-Standard entschlüsselt, DCTDecode = Bilddaten werden nach dem JPEG-Verfahren entpackt).

Anmerkung: Die /ASCIIxx-Filter werden benutzt, um binäre Bilddaten in ASCII-Zeichen zu überführen, da PDF-Dateien nur druckbare Zeichen enthalten dürfen. Beim ASCIIHexDecode-Filter wird das Ende der Zeichenfolge durch > markiert. Der ASCII85Decode-Filter erzeugt aus vier Byte Binärdaten jeweils fünf druckbare Zeichen. Die vier Bytes b1, b2, b3 und b4 werden dabei gemäß folgender Formel in die Zeichen c1, c2, c3, c4 und c5 konvertiert: *b1*256**3 + b2**2+b3* + b4 = c1*85**4 + c2*85**3 + c3*85**3 + c4*85**2 + c5*. Die fünf Ziffern c1 bis c5 werden durch Addition der Zahl 33 in druckbare Zeichen umgesetzt (Zeichen ! bis u). Bei fünf aufeinanderfolgenden Nullen wird das Zeichen *z* (anstelle von !!!!) als Platzhalter eingesetzt. Enthält der Rest des Streams keine vier Bytes mehr, sind die fehlenden Bytes mit Nullen aufzufüllen und dann zu kodieren. Anschließend werden aber nur die Ziffern gespeichert, deren Wert nicht 0 war. Das Ende der Datensequenz wird durch die Zeichen ~> markiert.

Eine detaillierte Beschreibung des PDF-Formats finden Sie in dem Adobe-Titel »PDF Reference Manual, Second Edition: Version 1.3«, erschienen bei Addison-Wesley (ISBN 0-201-61588-6).