Mark Stephens

Mark has been working with Java and PDF since 1999 and is a big NetBeans fan. He enjoys speaking at conferences. He has an MA in Medieval History and a passion for reading.

How are Embedded CMAP tables defined in a PDF File?

May 18, 2012 · 2 min read

Every glyf inside a PDF file can have a display value and a different extraction value. This is useful because often you need to display something different on the screen to what you get if you search the PDF file or extract the text. For example lignatures (**fl**, **fi**, etc) look much better when displayed using a special value rather than an **f** followed by an **l** or **i**. But when you search for **floor** or **fine**, you want these to be found correctly. So the PDF file format allows you to define separate values for display and actual text value.

One of the ways you can setup the values used for extraction is to use a CMAP table. It can be stored inside a PDF object (and often compressed). Lets see what one looks like if we extract the actual data...

```
/CIDInit /ProcSet findresource begin 12 dict begin begincmap
/CIDSystemInfo <<
/Registry (F6+0) /Ordering (T1UV) /Supplement 0 >> def
/CMapName /F6+0 def
/CMapType 2 def
1 begincodespacerange <02> <b7> endcodespacerange
19 beginbfchar
<07> <03C0>
<09> <0061>
<0a> <006D>
<0b> <0070>
<1e> <02DA>
<20> <0020>
<22> <0022>
<3d> <003D>
<3f> <003F>
<59> <0059>
<5b> <005B>
<5d> <005D>
<5f> <005F>
<7d> <007D>
<84> <2014>
<85> <2013>
<90> <2019>
<b0> <00B0>
<b7> <00B7>
endbfchar
8 beginbfrange
<24> <25> <0024>
<27> <29> <0027>
<2b> <2e> <002B>
<30> <3b> <0030>
<41> <50> <0041>
<52> <57> <0052>
<61> <7b> <0061>
<8d> <8e> <201C>
endbfrange
6 beginbfrange
<02> <02> [<0066006C>]
<03> <03> [<00540068>]
<04> <04> [<00660069>]
<05> <05> [<00660074>]
<06> <06> [<00660066>]
<08> <08> [<006600660069>]
endbfrange
endcmap CMapName currentdict /CMap defineresource pop end end
```

The first thing to note is that it is a readable text file (always much easier to follow!). The file has a header but the really interesting part for us is the lines between the begin/end tags. Note we can have multiple tables – in this file there are two **beginbfrange** sections and we read both in turn. All the values are hex with the unicode values being shown as 4 characters even if the high order byte is zero.

beginbfchar is the simpler of the two. It shows a section of single values with their unicode values. So character 7 maps onto unicode hex value 03C0 for the purpose of search and extraction, character 9 maps onto unicode hex value 0061, etc.

The **beginbfrange** sections allow you to specify a range of values to fill starting with a certain value (which is incremented for each one). This is what happens in the first section. So the line

<27> <29> <0027>

means 27 is mapped to 27, 28 is mapped to 28, 29 is mapped to 29

But we can also map a single character onto multiple values (allowing us to have a single character for **fl** or ... but set the correct values for the actual text). So the line

<02> <02> [<0066006C>]

means that character 2 is actually 2 text characters (unicode hex 66 and 6C). We use these for the text value, but the single glyf value defined in the font for display purposes.

So CMAP files are very useful because we can use them to provide very flexible options for what the text value of any character should be. *Are you using them in your PDF files?*

Fonts PDF

« Replacing the deprecated Java JPEG classes for Java 7

Avoid transparency when printing in Java »

How to insert an image into a PDF

Recently, we released JPedal 2023.07 which contains the ability to insert images into PDF files. All you need is a copy of JPedal, a...

Jacob Collins
Jul 7, 2023 · 18 sec read

What is tagged PDF?

Mark Stephens
May 19, 2023 · 52 sec read

What is the future of PDF forms?

Mark Stephens
May 17, 2023 · 1 min read