How to create and use custom fonts for PDF generation

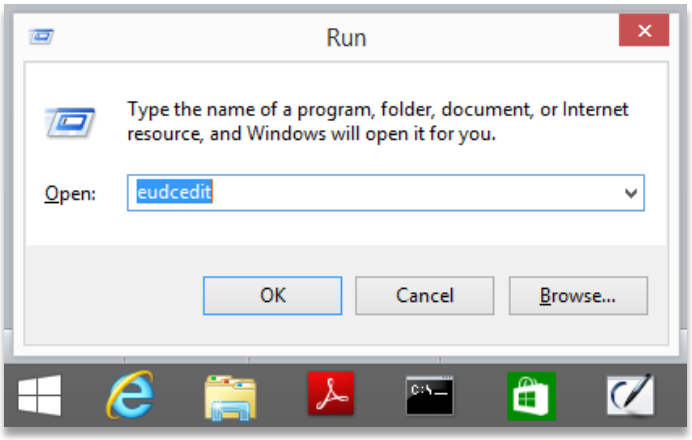
Written by Apitron Documentation Team

Introduction

Despite there are plenty of fonts created for various reasons sometimes you may want to create your own “custom” font, which would contain characters needed for particular reasons. Apitron PDF Kit makes you able to achieve this by using several possible ways described in subsequent sections.

Design custom TrueType font using built-in EUDC editor (Windows)

You may use any font editor capable of handling TrueType or OpenType fonts, and the most simplistic and available is end user character editor provided by Microsoft and shipped with Windows. It’s called *eudcedit* and one may run it by executing the **eudcedit** command in *Run* window (can be invoked by pressing Windows Key + R).

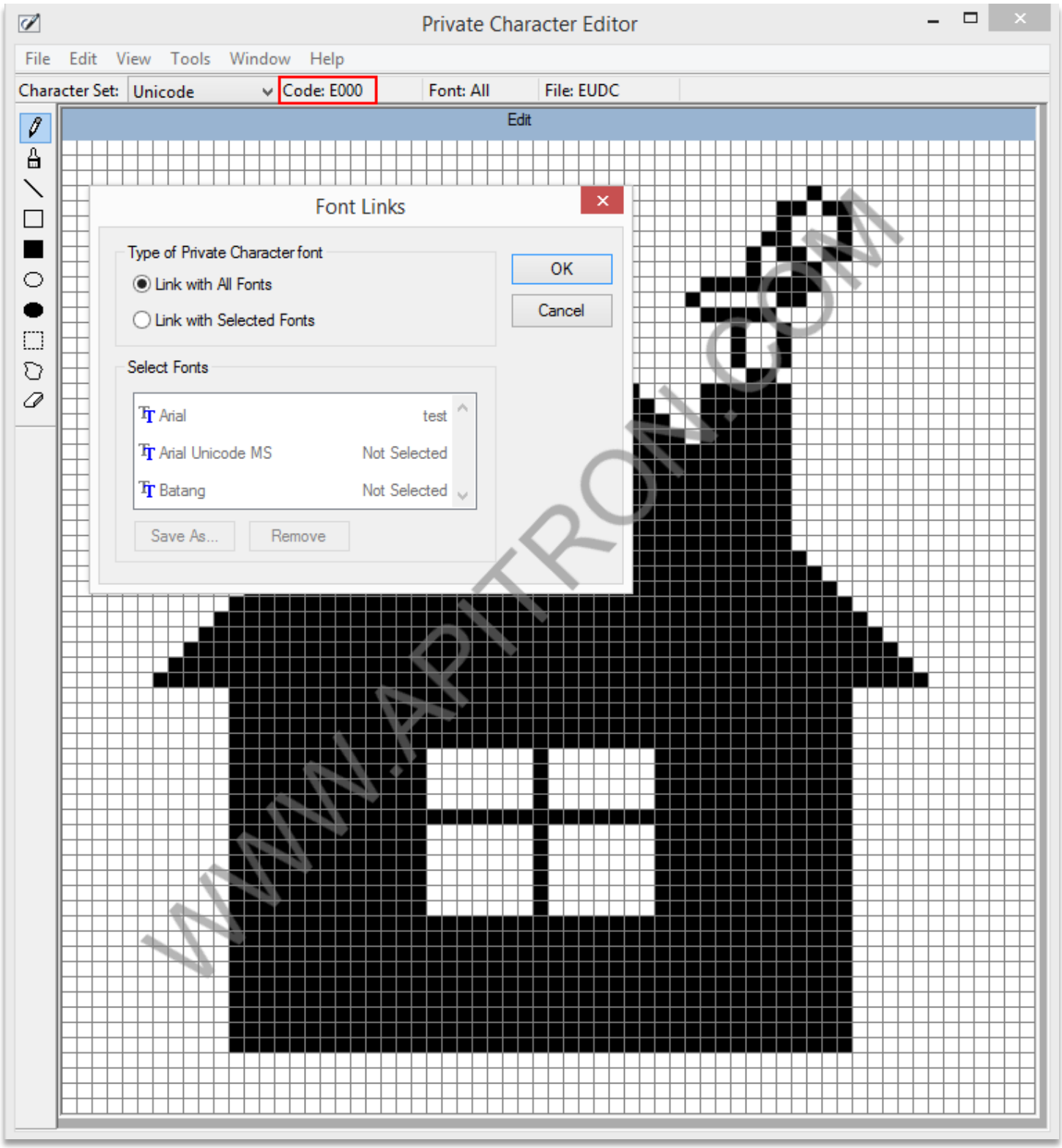


Pic. 1 Run command window

The editor starts and you can now design your own characters using its graphical tools and also define their codes. When you’re done with characters design you may then save them by choosing *File -> Font Links…* and pressing OK in invoked dialog.

From now on, these end user defined characters can be used by many text editing and viewing applications if they implement the EUDC support. If they use Windows API for displaying text it will work automatically.

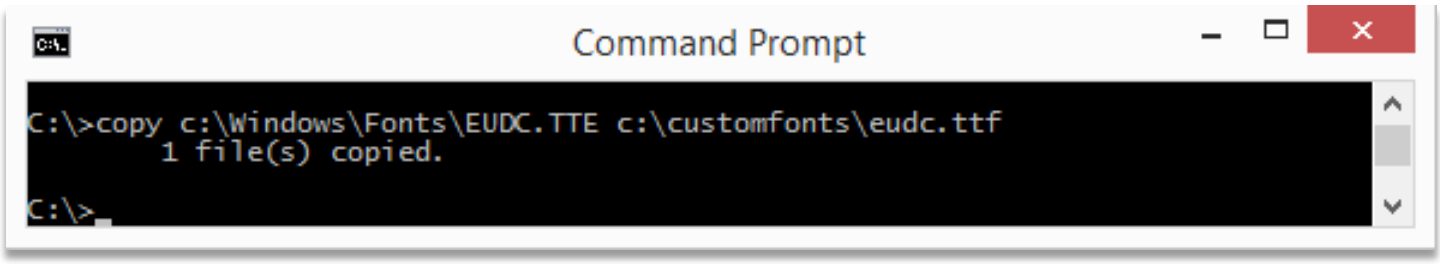
We are going to design a few simple characters, and use the resulting font in PDF generation routine. See the image below showing the character editor window, code assigning and font saving process.



Pic. 2 Designing character using EUDC editor

The assigned character code was highlighted on the preceding image, we used the one assigned by default but it's possible to use any of the allowed codes. Let's use the designed and saved character and create a PDF file containing it.

For it to work let's copy the resulting font from windows fonts directory and rename the resulting file by changing its extension from *tte* to *ttf*. Assuming that OS is installed at "*C:\Windows*" and we are going to copy the file to "*C:\customfonts\eudc.ttf*" it looks as follows (using command prompt).

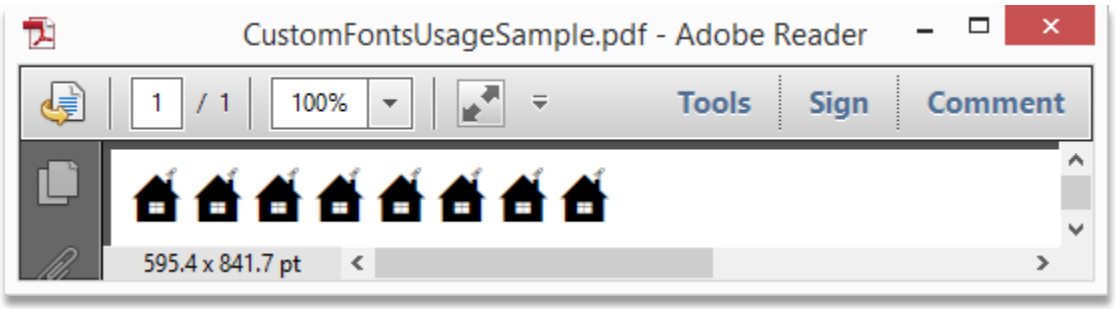


Pic. 3 Copying EUDC font

The following code produces PDF file containing the characters we created:

```
// create document
FlowDocument document = new FlowDocument() { Margin = new Thickness(5) };

// lets create the village, using \uE000 charcode for each symbol
document.Add(new TextBlock("\uE000\uE000\uE000\uE000\uE000\uE000\uE000\uE000")
                { Font = new Font(@"c:\customfonts\eudc.ttf", 20) });

// generate PDF document
using (Stream stream = File.Create("CustomFontsUsageSample.pdf"))
{
    document.Write(stream, new ResourceManager(), new PageBoundary(Boundaries.A4));
}
```

The image below demonstrates the result:



Pic. 4 Generated PDF file

Furthermore, this custom font will be embedded into the resulting document making it viewable on other devices which obviously don't have our custom font installed.

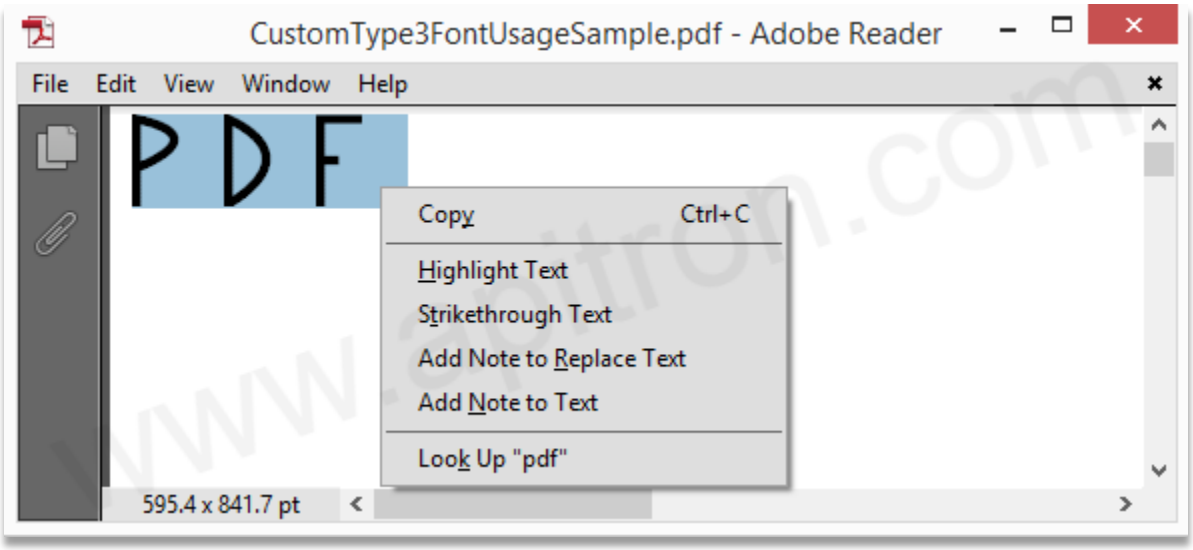
Create Type3 PDF fonts using Apitron PDF Kit API

PDF allows you to use any sequence of drawing commands as a “glyph” and therefore combine these glyphs in so-called Type3 fonts. Apitron PDF Kit provides API for creation of such fonts and the code below shows how to create several Type3 glyphs and uses them to produce text.

```
// define first glyph as P letter
Type3FontGlyph glyph1 = new Type3FontGlyph('p', 20,20);
Path path1 = new Path(1,0);
path1.AppendLine(1, 19);
path1.AppendCubicBezierUsingCurrentPoint(20, 15, 0, 8);
glyph1.SetLineWidth(2);
glyph1.StrokePath(path1);

// define second glyph as D letter
Type3FontGlyph glyph2 = new Type3FontGlyph('d', 20, 20);
Path path2 = new Path(1,0);
path2.AppendLine(1, 19);
path2.AppendCubicBezierUsingCurrentPoint(20, 10, 0, 0);
glyph2.SetLineWidth(2);
glyph2.StrokePath(path2);

// define third glyph as F letter
Type3FontGlyph glyph3 = new Type3FontGlyph('f', 20, 20);
Path path3 = new Path(1,0);
path3.AppendLine(1, 19);
path3.AppendLine(10, 19);
path3.MoveTo(1,10 );
path3.AppendLine(5, 10);
glyph3.SetLineWidth(2);
glyph3.StrokePath(path3);

// create type3 font object
Type3Font type3Font = new Type3Font("myFont");

// add glyphs to font object
type3Font[glyph1.Unicode] = glyph1;
type3Font[glyph2.Unicode] = glyph2;
type3Font[glyph3.Unicode] = glyph3;

// create resource manager and register font
ResourceManager resourceManager = new ResourceManager();
resourceManager.RegisterResource(type3Font);

// create document
FlowDocument document = new FlowDocument() { Margin = new Thickness(5) };

// append text
document.Add(new TextBlock("pdf"){Font = new Font("myFont", 1)});

// generate PDF document
using (Stream stream = File.Create("CustomType3FontUsageSample.pdf"))
{
    document.Write(stream, resourceManager, new PageBoundary(Boundaries.A4));
}
```

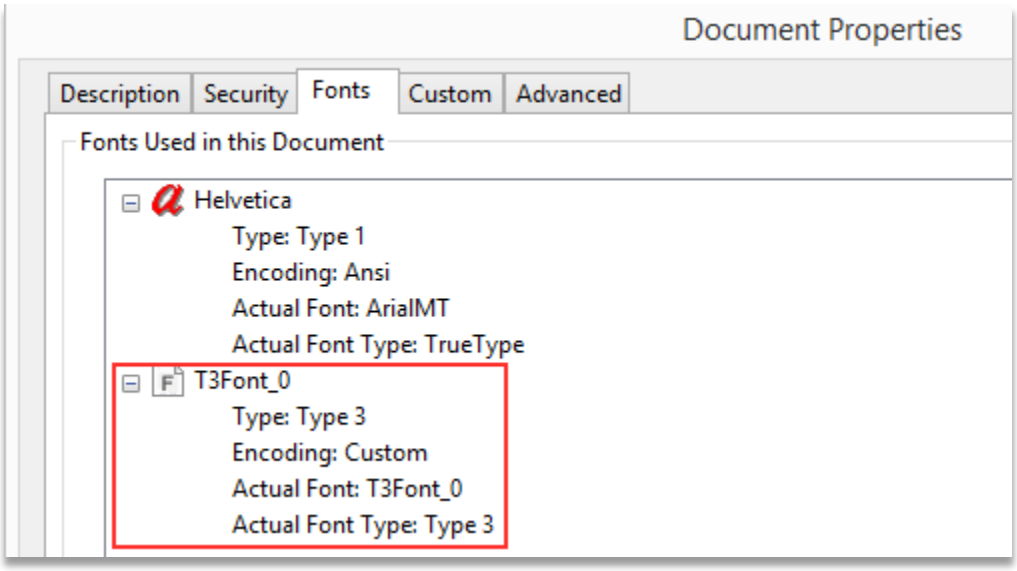
As it can be seen from code on previous page one may define any graphical shape as a glyph by using PDF drawing commands and create Type3 font from a set of such glyphs which can be further used to write text.

The resulting PDF file is shown on the image below:



Pic. 5 Text written using Type3 font

You may notice that this text is perfectly selectable in PDF viewer and can be copied and pasted to anywhere (see the lowest right click menu option offering to search for the text “pdf”). The Type3 font used will be embedded in PDF file and so it will be viewable on all systems. See the file properties dialog, fonts tab.



Pic. 6 Fonts used in generated PDF file