PDFtext.dll

You can use it with all well known ides (.NET-IDEs, too!)
Try it with Visual Basic, VBA, C#, VB200x, VB200x Express, Delphi, C, C++, ...

This is the test version!
This means "try before you buy"!
If you think this dll can be useful for your work please order the unlimited version at:
www.PDF-Analyzer.com
This version here isn't limited in any cases so you're able to test the whole functionality.
There's only one difference to the full version:
The message window calling a function ;-)

You can get the unlimited version from:
Ingo Schmoekel
- Software-Dev. and Distribution -
Zedernstr. 30a
D-28832 Achim
GERMANY

Webmaster@PDF-Analyzer.com
http://www.PDF-Analyzer.com
http://www.IS-Soft.de

----- Index -----

What are the limits:
The textextraction works with all types of normal pdf-documents from pdf-specs 1.2 up to the new specs 1.7. It doesn't mind if there's an AES- or RC4-encryption ... nothing, 40 or 128 bit ... even a main-/owner-password isn't a problem!
What you can't extract are ebooks (they've a special protection), pdf-documents protected with a user-password (the one you have to insert before reading the document-content on the screen) and real pictures converted to pdf without ocr-functionality 'cause there isn't real text.

About installation and working with the dll...
This dll doesn't have an entry point so it's not necessary to register it.
Only copying into the system32-, SysWOW64-directory or in the directory where your application is installed.

Working inside an IDE:
→If you're working inside the IDE from VB 6.0 the file PDFtext.dll should be in the VB98-directory 'cause then the application is VB 6.0 and not the project you're testing.
→If you're working inside the IDE from VB 2005 Express the file PDFtext.dll should be in the bin-directory (under your project-directory).
→If you're using vba (from ms access or excel for example) you should declare the functions as private and not public.
→Using .NET-environments (like MS Visual Studio) it's often better (to avoid stack imbalance errors) to change the parameter-types from longinteger to integer or int32 /vb2005).

If you want to use it in an asp environment...
Using for example Windows 2003 you should have a look at the "Data Execution Prevention (DEP)". The "DEP" should be disabled for the process using the PDFtext.dll!
How to do it:
Control panel → system → advanced → power → settings → DEP.
Here you should add PDFtext.dll/the "IIS Worker Process". At the end you have to perform an IISreset at the command prompt. That's all ... in this way the PDFtext.dll can work in an asp environment (for example with Windows 2003).

If you want to use it in a 64 bit environment...
PDFtext.dll is a 32 bit dll but you can use it in 64 bit environments, too. For example on Win XP Pro x64 the calling application should be installed in the "program files (x86)" directory and it'll work.
If you want to insert the dll in the system32-directory: This directory is used only for 64 bit dlls if it's a 64 bit system. In this case the PDFtext.dll (= 32 bit) should be in the SysWOW64-directory.
If you're working with a 64 bit development you may need to set your platform target to "x86" to force it to create a 32 bit EXE. If it's currently set to "Any CPU” then it could be creating a 64 bit exe which then will fail loading a 32 bit dll.

If you're using a win-version before xp or win7...
It's possible that the needed 32-bit-dll gdiplus.dll is missing...
If so, your app (working with PDFtext.dll) sometimes could deliver faulty results.
To avoid this as a standard the gdiplus.dll is in the zip-package and should be copied where the PDFtext.dll will be installed (for example in your app-directory).

Something about textextraction
The text will be extracted for each page like it was composed (if you use "fast = 0"). So if each page have got first a header and a footer, this content will be extracted for example before the first headline of the page. If a page has several columns and if these columns were inserted top down you'll find for example the last line of the first column before the

first textline of the second column. If you like it more as it's displayed in the pdf-document (it's faster, too) you can use "fast = 1".

If you're using the parameter "stop" you can't be sure that the extraction stops after the seconds you've set. For example you've set to stop after 4 seconds and after 3 seconds the extraction for a very complicated page is running then you've to wait until the next page comes. In this described case the extraction can stop after 10 or more seconds, too.

The content of the zip-file

The library itself as a testversion
PDFtext.dll (in the windows\system32-, syswow64- or in your application-directory)

The short documentation
H_PDFtext.pdf (the file you’re just reading)

For the library there is a help-program to test the function immediately:
H_PDFtext.exe (a version made with Delphi 2007)
H_PDFtextFPL.exe (a version made with Lazarus/Free Pascal)

As source example there is a Delphi-, Free Pascal-/Lazarus-, VB- and VB2005 Express-project included:
H_PDFtext_sources_FreePascalLazarus.zip
H_PDFtext_sources_Delphi5_Delphi2007.zip
H_PDFtext_VB.zip
H_PDFtext_VB2005Express.zip
Pdf_sample_with_ms_cpp.zip (a common sample calling my dlls via MS C++)

contact: webmaster@pdf-analyzer.com
info/help: http://www.pdf-analyzer.com
http://www.is-soft.de

Ingo Schmoekel
- Software-Dev.& Distribution -
Zedernstr.30a
D-28832 Achim - Uesen
GERMANY

Kinds of returned (error) codes for GetPDFPageCount:

9001 = File not found
9002 = No pdf-file
9010 = PDF-file couldn't processed
9013 = The pdf has a structure problem ... the creation wasn't proper
9014 = The pdf has a problem with the dictionary ... the creation wasn't proper
If it's all okay the pagecount of the selected document will be returned.

Kinds of returned (error) codes for GetPDFText:

9001 = File not found
9002 = No pdf-file
9005 = Target path isn't valid
9010 = PDF-file couldn't processed
9011 = There's a user password ... no textextraction is possible
9012 = No access to user- or app-path for the working-version of the pdf-file

9013 = The pdf has a structure problem ... the creation wasn't proper
9014 = The pdf has a problem with the dictionary ... the creation wasn't proper
9015 = The text is based on the rare codepage 1251 ... extraction won't work proper
9016 = The text is based on the codepage CJK ... extraction won't work proper
1 = For option 2 ... means it's okay.
Option 1 returns the address of the textoutput.
Option 3 returns the whole text-string.
9 = Means that there's no text (perhaps only images)
If you extract with success into a textfile (opt=1) as a result you'll get the complete address of the textoutput. So it's easy to work with the file programmatically.

functions with the type of values and the meaning:

GetPDFPageCount

function GetPDFPageCount(const FileName: PChar): LongInt; stdcall;

GetPDFText

function GetPDFText(const FileName: PChar; opt: LongInt; hw: LongInt; fast: LongInt; target: PChar; lspaces: LongInt; ptitel: PChar; pos: LongInt; page: LongInt; clock: LongInt; blank: LongInt; ende: LongInt; wlist: LongInt): PChar; stdcall;

FileName
That's the pdf-file and can't be empty ;-)

opt
opt=1 means extract the text-content of example.pdf to example.pdf.txt (in the directory where's the dll is or in the open user-directory if exist). User-directory normally means "documents and settings\the-login-name\local settings\temp\pdftext".
opt=2 means extract the text-content of a pdf-file to the clipboard.
opt=3 means extract the text-content of a pdf-file as a returning text-string.

hw (default = 0)
If this value is set to "1" only a hash-value (md5) will be returned. You can use it to check if the text-content is changed between two checks.

fast (default = 1)
Fast means that the text-content of a page will be extracted like it is. If you set this parameter to zero the text-content will be extracted like it was inserted. If line 3 was inserted before line 1, you'll see line 3 first... To get all these text parts separately last a bit longer.

target
If you want to extract into a textfile (opt=1) you should fill "target" with a new filename (with drive and path!). If you don't do this the file will be created in the free userpath (local settings -> temp -> pdftext_temp{...}.pdf.txt).

lspaces (default = 0)
lspaces means 1 to delete leading spaces on each text-line or 0 (don't do it).

ptitel (default = '')
ptitel means that pagenumbering should be inserted. If you insert here for example "page", the pagenumbering would be "page x / y". If you insert "Seite" it would be "Seite x / y".

pos (default = 0)
1 means that all extracted textstrings will have there row and column (in pixels). So you can get exactly the string position on a page. Keep in mind that the highest row number is at the top of the page. This parameter works only if "fast" is set to zero.

page (default = 0)
If you don't want to extract the whole text of a document... if you have special known pages you want to extract, you can insert in "page" single page numbers. Then only these page will be extracted.

clock (default = 0)
0 means no sandclock while the function is working...

blank (default = 0)
There are documents with justification layout. This means that sometimes (to fill each line) there are more than one space between words. If you're using PDFtext.dll to search in a second step through extracted textcontent it could be easier if you know that between words are only one space. If "blank" is set to 1 more than one space between words will be deleted.

ende (default = 0)
Sometimes you can get very voluminous documents with much different contents - The extraction can last many seconds... minutes. Sometimes it's enough if you've extracted the first pages... Here you can set a value that means "stop extracting after xxxx seconds". 0 means no limit, 1 until 1800 means from 1 second until 30 minutes (that should be enough).

wlist (default = 0)
wlist means wordlist. If this parameter is set to "1". The extraction delivers word by word. Each word on a separate row.

Sample for delphi: Including the dll in a delphi unit

```
unit H_PDFtext;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;

type
  TForm1 = class(TForm)

// . . .

function GetPDFText(const FileName: PChar; opt: LongInt; hw: LongInt; fast:
LongInt; target: PChar; lspaces: LongInt; ptitel: PChar; pos: LongInt;
page: LongInt;

clock: LongInt; blank: LongInt; ende: LongInt; wlist: LongInt): PChar;
stdcall;

function GetPDFPageCount(const FileName: PChar): LongInt; stdcall;

implementation

{$R *.DFM}

function GetPDFText(const FileName: PChar; opt: LongInt; hw: LongInt; fast:
LongInt; target: PChar; lspaces: LongInt; ptitel: PChar; pos: LongInt;
page: LongInt;

clock: LongInt; blank: LongInt; ende: LongInt; wlist: LongInt): PChar;
stdcall;
external 'PDFtext.dll';

function GetPDFPageCount(const FileName: PChar): LongInt; stdcall;
external 'PDFtext.dll';

// . . .

procedure TForm1.Button1Click(Sender: TObject);
begin
     If OpenDialog1.Execute Then
        Edit1.Text := OpenDialog1.FileName;
end;

// . . .

procedure TForm1.Button2Click(Sender: TObject);
var s  : LongInt;
    pt : String;
    sp : LongInt; // no leading spaces
    pa : LongInt; // extract only page number ...
    po : LongInt; // with positions
    st : LongInt; // stop after ... seconds
    fa : LongInt; // fast
    cl : LongInt; // sandclock on
    mp : LongInt; // no multiple spaces
begin
     if ( CheckBox1.Checked = True ) then
        sp := 1
```

```
       else
        sp := 0;

     if ( CheckBox2.Checked = True ) then
        po := 1
       else
        po := 0;

     if ( CheckBox3.Checked = True ) then
        fa := 1
       else
        fa := 0;

     if ( CheckBox4.Checked = True ) then
        cl := 1
       else
        cl := 0;

     if ( CheckBox5.Checked = True ) then
        mp := 1
       else
        mp := 0;

     if ( Trim(Edit4.Text) <> '' ) Then // pagetitle
        pt := Edit4.Text;

     if ( Trim(Edit6.Text) <> '' ) Then // stop after ... seconds
        begin
           Edit6.Text := Trim(Edit6.Text);
           st := StrToInt(Edit6.Text);
        end;

     pa := StrToInt(Trim(Edit5.Text)); // extract only page number ...

//   opt=1 means extract to file ...
     Edit2.Text := GetPDFText(PChar(Edit1.Text), 1, 0, fa,
PChar(Edit3.Text), sp,
                   PChar(pt), po, pa, cl, mp, st, 0);

end;

procedure TForm1.Button3Click(Sender: TObject);
begin
     Edit3.Text := IntToStr(GetPDFPageCount(PChar(Edit1.Text)));
end;

// . . .

end.
```

How to use the dll with Delphi in the dynamic way

```
. . .
function PDFDoc.PdfTextGet(File: String): String;
var
  GetPdfText: TGetPdfText;
  FuncPtr: TFarProc;
  DLLHandle: THandle;
begin
  // dynamic load for the dll and textextraction
  DLLHandle := LoadLibrary(PChar('PDFtext.dll'));
  FuncPtr := GetProcAddress(DLLHandle, 'GetPDFText');
  @GetPdfText := FuncPtr;
  Result := GetPdfText(PChar(File), 3, 0, PChar(''), 0, Pchar(''),0,0,0,
0);
  FuncPtr := nil;
  FreeLibrary(DLLHandle);
end;
. . .
```

Sample for visual basic: Including the dll in a bas-modul module1

```
. . .
Public Declare Function GetPDFText Lib "PDFtext.dll" (ByVal FileName As
String, ByVal opt As Long, ByVal hw As Long, ByVal fast As Long, ByVal
target As String, ByVal

xlspaces As Long, ByVal ptitel As String, ByVal pos As Long, ByVal page As
Long, ByVal clock As Long, ByVal blank As Long, ByVal ende As Long, ByVal
wlist As Long) As

String

Public Declare Function GetPDFPageCount Lib "PDFtext.dll" (ByVal FileName
As String) As Long

. . .
and two of the four button-clicks for the options:
. . .
Private Sub option1_Click()
  Dim fa As Integer 'fast or structured extraction
  Dim po As Integer 'with string-positions
  Dim sp As Integer 'no leading spaces
  Dim pa As Integer 'extract only page number ...
  Dim cl As Integer 'showing sandclock
  Dim mp As Integer 'no multiple spaces between words
  Dim st As Integer 'stop after ... seconds

    If Check1.Value = 1 Then
       sp = 1
      Else
       sp = 0
    End If
    If Check2.Value = 1 Then
       po = 1
      Else
       po = 0
    End If
    If Check3.Value = 1 Then
       fa = 1
      Else
       fa = 0
    End If
    If Check4.Value = 1 Then
       cl = 1
      Else
       cl = 0
    End If
    If Check5.Value = 1 Then
       mp = 1
      Else
       mp = 0
    End If

    pa = Val(Trim(Text5.Text))
    st = Val(Trim(Text6.Text))

'   *** 1 = Extract to file ... ***
'   *** 2 = Extract to clipboard ... ***
'   *** 3 = Extract to/as string ... ***
```

```
    Text2.Text = GetPDFText(Text1.Text, 1, 0, fa, Text3.Text, sp,
Trim(Text4.Text),
                 po, pa, cl, mp, st, 0)
End Sub

Private Sub option2_Click()
    Text3.Text = GetPDFPageCount(Text1.Text)
End Sub

Private Sub Text5_LostFocus()
   If (Val(Trim(Text5.Text)) > 99999) Or _
      (Val(Trim(Text5.Text)) < 0) Then
      Text5.Text = "0"
   End If
End Sub

Private Sub Text6_Change()
   If (Val(Trim(Text6.Text)) > 1800) Or _
      (Val(Trim(Text6.Text)) < 0) Then
      Text6.Text = "0"
   End If
End Sub
. . .
```

Sample for C++

```
. . .
typedef LPSTR (__stdcall *TRMyGetPDFText) (LPSTR strPdfName,int opt,int
hw,int
fast,LPSTR target,int lspaces,LPSTR ptitel,int pos,int page,int clock,int
blank, int ende, int wlist);
TRMyGetPDFText    GetPDFText;


Main()
{
      m_hInstLib = LoadLibrary( (LPCSTR)"PDFText.dll" );
      if( m_hInstLib )
      {
            CString s;
            GetPDFText = (TRMyGetPDFText) GetProcAddress( m_hInstLib,
"GetPDFText");


s=GetPDFText("c:\\test.pdf",1,0,0,"c:\\test.txt",0,0,1,0,0,0,0,0);
            FreeLibrary(m_hInstLib);
      }
}
. . .
```