PDFtext.dll – 32 bit-version – 4.0.0.0

You can use it with all well known ides (.NET-IDEs, too!)
Try it with Visual Basic, VBA, C#, VB20xx, VB20xx Express, Delphi, C, C++, PowerBuilder and many more …

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. It opens once calling a function ;-)

You can get the unlimited version from:

Author and publisher:
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

credits:
Thanks to Heiko Indenbirken for a C#-sampleclass "PDFText_32bit_or_64bit.cs"
Thanks to Nicholas Vollmer for a vb.net-sampleclass "class_for_PDFtext40_Form1.vb"
Both in the zip-package!

If your question begins with "how can I …" please have a quick look into the complete sample projects (for 32 bit) in this package ;-)

----- Index -----

What are the limits:
The textextraction works with all types of normal pdf-documents from pdf-specs 1.2 up to the newest specs. It doesn't mind if there's an AES- or RC4-encryption ... nothing, 40, 128 or 256 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- (on 32-bit-systems), SysWOW64-directory (on 64-bit-systems) or in the directory where your application is installed. If you want to use the dll with vba inside MS Access, Excel, … then the dll has to be in the system32 or syswow64 (application directory won't work). Choosing the dll inside for example a mdb-project via extras -> references won't work 'cause the dll can't be registered without an entry point – so simply copying into system32 or syswow64 … that's all.

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: On 64 bit systems this directory is used only for 64 bit dlls. 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 will create a 64 bit exe which then will fail loading this 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/copied.

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. To make it more readable the string-output is sorted (top/down and left/right). If you like it more similar to the original layout 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 … a 32-bit-dll for the windows\system32-, syswow64- or in your application-directory (if it's not a vba- or .net-environment)

The short documentation
H_PDFtextsv32.pdf (the file you're just reading)

For the library there is a help-program to test the function immediately:
H_PDFtext.exe (a 32-bit-version made with Delphi XE as 32 bit version)

As source samples there are codesnippets in Delphi- (static/dynamic), Free Pascal-/Lazarus-, VB-, C#- and C++-code at the end of this document here.

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 regarding GetPDFPageCount:

0	= general/main error
9001	= File not found
9002	= No pdf-file
9003	= There's a user password
9004	= Invalid/damaged page structure

If it's all okay the pagecount of the selected document will be returned.

Kinds of returned (error) codes regarding GetPDFText:

0	= general/main error
9001	= File not found
9002	= No pdf-file
9003	= There's a user password
9004	= Invalid/damaged page structure
9005	= Target drive/path isn't valid
9006	= Target drive/path is missing
9007	= Source and target (for fileoutput) is the same
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	= Using opt=2 (clipboard) ... this means it's okay. Using opt=1 returns the drive/path/file of the textoutput. Using opt=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: PWideChar):
LongInt; stdcall;

GetPDFText

function GetPDFText(const FileName: PWideChar; opt: LongInt; hw: LongInt; fast: LongInt; target: PWideChar; lspaces: LongInt; ptitel: PWideChar; pos: LongInt; page: LongInt; clock: LongInt; blank: LongInt; ende: LongInt; wlist: LongInt): PWideChar; 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 similar to the original pdf-layout. If you set this parameter to zero the text-content will be extracted string by string and sorted.

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 error 9006 will raise – telling you that the target data is missing.

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 will look like "page x / y". If you insert "Seite" it would be "Seite x / y". If ptitel is empty there won’t be a pagenumbering on the extracted textcontent.

pos (default = 0)
1 means that all extracted textstrings (fast=0) will have four leading values: current pagenumber, max. pagenumber, current 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 bottom 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 XE (Unicode!): Including the dll in a delphi unit

There's **a complete "ready-to-run" Delphi-project** in this zip-Package!

```
unit H_PDFtext;

interface

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

type
  TForm1 = class(TForm)

// . . .

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

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

implementation

{$R *.DFM}

function GetPDFText(const FileName: PWideChar; opt: LongInt; hw:
LongInt; fast: LongInt; target: PWideChar; lspaces: LongInt; ptitel:
PWideChar; pos: LongInt; page: LongInt; clock: LongInt; blank: LongInt;
ende: LongInt; wlist: LongInt): PWideChar; stdcall;
external 'PDFtext.dll';

function GetPDFPageCount(const FileName: PWideChar): 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  txtw : WideString;
     txtp : PWideChar;
//    . . .

//   opt=1 means extract to file ...
     Edit2.Text := GetPDFText(PWideChar(Edit1.Text), 1, hw, fa,
                   PWideChar(Edit3.Text), sp, PWideChar(pt), po, pa, cl,
                   mp, st, wl);
//    … or …
//   opt=3 means extract to a returning string ...
//   In this case it is absolutely necessary to keep in mind that the
//   dll-parameters are pointers to a string!
     txtp := GetPDFText(PWideChar(Edit1.Text), 3, hw, fa,
             PWideChar(Edit3.Text), sp, PWideChar(pt), po, pa, cl,
             mp, st, wl);
     txtw := WideCharToString(txtp);
     Edit2.Text := txtw;
end;

procedure TForm1.Button3Click(Sender: TObject);
begin

//   . . .

     Edit3.Text := IntToStr(GetPDFPageCount(PWideChar(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(PWideChar (File), 1, 0, PWideChar (''), 0,
            Pchar(''),0,0,0, 0);
  FuncPtr := nil;
  FreeLibrary(DLLHandle);
end;
. . .
```

Sample for Visual Basic 6.0/VBA: Including the declared dll-functions in a bas-modul (if it's real Visual Basic) module1

There's **a complete "ready-to-run" vb6-project** in this zip-Package!

```
. . .

Attribute VB_Name = "Module1"

Public Declare Function GetPDFPageCount Lib "PDFtext.dll" (ByVal
FileName As String) As Integer  ' Pagecount
Public Declare Function GetPDFText Lib "PDFtext.dll" (ByVal FileName As
String, ByVal opt As Integer, ByVal hw As Integer, ByVal fast As Integer,
ByVal target As String, ByVal xlspaces As Integer, ByVal ptitel As String,
ByVal pos As Integer, ByVal page As Integer, ByVal clock As Integer, ByVal
blank As Integer, ByVal ende As Integer, ByVal wlist As Integer) As Long '
The returned text content
```

```
Public Declare Function apiLStrCopyW Lib "kernel32.dll" Alias "lstrcpyW"
(ByVal lpString1 As Long, ByVal lpString2 As Long) As Long
Public Declare Function apiLStrLenW Lib "kernel32.dll" Alias "lstrlenW"
(ByVal lpString As Long) As Long

Public Function GetStringFromPtrW(ByVal ptr As Long) As String
  'create a matching buffer
  GetStringFromPtrW = String$(apiLStrLenW(ptr), 0)
  'copying the string into the buffer
  apiLStrCopyW StrPtr(GetStringFromPtrW), ptr
End Function

. . .
```

And a button-click for GetPDFPageCount:

```
`. . .

Private Sub Command1_Click()
  Dim sPfad() As Byte

    sPfad = StrConv(Text1.Text, vbUnicode)

    Text7.Text = Str(GetPDFPageCount(sPfad))

End Sub

`. . .
```

and a button-click for string-extract:

```
`. . .

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

Private Sub option3_Click()
  Dim sPfad() As Byte
  Dim tPfad() As Byte
  Dim title() As Byte
  Dim sp As Integer
  Dim hv As Integer
  Dim po As Integer
  Dim pa As Integer
  Dim st As Integer
  Dim fa As Integer
  Dim cl As Integer
  Dim bl As Integer
  Dim wl As Integer

    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
       bl = 1
      Else
       bl = 0
    End If
    If Check6.Value = 1 Then
       hv = 1
      Else
       hv = 0
    End If
    If Check7.Value = 1 Then
       wl = 1
      Else
       wl = 0
    End If

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

    sPfad = StrConv(Text1.Text, vbUnicode)
    tPfad = StrConv(Text3.Text, vbUnicode)
    title = StrConv(Text4.Text, vbUnicode)

    Text2.Text = GetStringFromPtrW(GetPDFText(sPfad, 3, hv, fa, tPfad,
sp, title, po, pa, cl, bl, st, wl))

End Sub

' . . .
```

How to use the dll with vb.net

There's **a complete "ready-to-run" 32-bit-project** in this zip-Package! Additionally a well documented vb.net-sample called "class_for_PDFtext40_Form1.vb" is attached!

```
. . .

In Module1.vb …

Option Strict Off
Option Explicit On
Module Module1

    Public Declare Function GetPDFPageCount Lib "PDFtext.dll" (ByVal
FileName As Byte()) As Short ' Pagecount
    Public Declare Function GetPDFText Lib "PDFtext.dll" (ByVal
FileName As Byte(), ByVal opt As Short, ByVal hw As Short, ByVal fast As
Short, ByVal target As Byte(), ByVal xlspaces As Short, ByVal ptitel As
Byte(), ByVal pos As Short, ByVal page As Short, ByVal clock As Short,
ByVal blank As Short, ByVal ende As Short, ByVal wlist As Short) As Integer
' The returned text content

End Module

. . .

In Form1.vb …

Option Strict Off
Option Explicit On
Imports System.Runtime.InteropServices
Friend Class Form1
	Inherits System.Windows.Forms.Form
	Public r As String

    Private Sub Command1_Click(ByVal eventSender As System.Object, ByVal
eventArgs As System.EventArgs) Handles Command1.Click
        Dim uni_enc As New System.Text.UnicodeEncoding()
        Dim sPfad() As Byte

        sPfad = uni_enc.GetBytes(Text1.Text)

        Text7.Text = Str(GetPDFPageCount(sPfad))

    End Sub

' and now a button-click for string-extract

' . . .

    Private Sub option3_Click(ByVal eventSender As System.Object, ByVal
eventArgs As System.EventArgs) Handles option3.Click
        Dim uni_enc As New System.Text.UnicodeEncoding()
        Dim tmpMemPath As IntPtr
        Dim objFso As Object
        Dim objFile As Object
        Dim mem As Integer
        Dim sPfad As Byte()
        Dim tPfad As Byte()
        Dim title As Byte()
        Dim sp As Short
        Dim hv As Short
        Dim po As Short
        Dim pa As Short
        Dim st As Short
        Dim fa As Short
        Dim cl As Short
        Dim bl As Short
        Dim wl As Short

        objFso = CreateObject("Scripting.FileSystemObject")
        objFile = objFso.GetFile(Text1.Text)
        mem = objFile.Size

        If Check1.CheckState = 1 Then
            sp = 1
        Else
            sp = 0
        End If
        If Check2.CheckState = 1 Then
            po = 1
        Else
            po = 0
        End If
        If Check3.CheckState = 1 Then
            fa = 1
        Else
            fa = 0
        End If
        If Check4.CheckState = 1 Then
            cl = 1
        Else
            cl = 0
        End If
        If Check5.CheckState = 1 Then
            bl = 1
        Else
            bl = 0
        End If
        If Check6.CheckState = 1 Then
            hv = 1
        Else
            hv = 0
        End If
        If Check7.CheckState = 1 Then
            wl = 1
        Else
            wl = 0
        End If

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

        sPfad = uni_enc.GetBytes(Text1.Text)
        tPfad = uni_enc.GetBytes(Text3.Text)
        title = uni_enc.GetBytes(Text4.Text)

        tmpMemPath = Marshal.AllocHGlobal(mem)
        tmpMemPath = GetPDFText(sPfad, 3, hv, fa, tPfad, sp, title, po,
pa, cl, bl, st, wl)
        Text2.Text = Marshal.PtrToStringUni(tmpMemPath)
        tmpMemPath = IntPtr.Zero

    End Sub
```

How to use the dll with C#

There's **a complete (ready-to-run) 32-bit-project** in this package. Additionally there's also a well documented C#-sample called "PDFText_32bit_or_64bit.cs" in this zip-package!

C#-sample - GetPDFPageCount and GetPDFText on a button-click

From Module1.cs the declarations …

```
using Microsoft.VisualBasic;
using Microsoft.VisualBasic.Compatibility;
using System;
using System.Collections;
using System.Data;
using System.Diagnostics;
using System.Drawing;
using System.Windows.Forms;
using System.Runtime.InteropServices;
namespace Projekt1
{
      static class Module1
      {
            [DllImport("PDFtext.dll", CharSet = CharSet.Ansi, SetLastError
= true, ExactSpelling = true)]

            public static extern short GetPDFPageCount(byte[]
FileName);
            [DllImport("PDFtext.dll", CharSet = CharSet.Ansi, SetLastError
= true, ExactSpelling = true)]
            // Pagecount
            public static extern int GetPDFText(byte[] FileName, short
opt, short hw, short fast, byte[] target, short xlspaces, byte[] ptitel,
short pos, short page, short clock,
            short blank, short ende, short wlist);
            // The returned text content

      }
}

` . . .
```

From Form1.cs the events …

C# - GetPDFPageCount on a button-click

```
namespace Projekt1
{
      internal partial class Form1 : System.Windows.Forms.Form
      {
            public string r;

            private void Command1_Click(System.Object eventSender,
System.EventArgs eventArgs)
            {
                  System.Text.UnicodeEncoding uni_enc = new
System.Text.UnicodeEncoding();
                  byte[] sPfad = null;

                  sPfad = uni_enc.GetBytes(Text1.Text);

                  Text7.Text =
Conversion.Str(Module1.GetPDFPageCount(sPfad));

            }

` . . .
```

C# - GetPDFText (as string export) on a button-click

```
            private void option3_Click(System.Object eventSender,
System.EventArgs eventArgs)
            {
                  System.Text.UnicodeEncoding uni_enc = new
System.Text.UnicodeEncoding();
                  IntPtr tmpMemPath = default(IntPtr);
                  int mem = 0;
                  byte[] sPfad = null;
                  byte[] tPfad = null;
                  byte[] title = null;
                  short sp = 0;
                  short hv = 0;
                  short po = 0;
                  short pa = 0;
                  short st = 0;
                  short fa = 0;
                  short cl = 0;
                  short bl = 0;
                  short wl = 0;

            FileInfo f = new FileInfo(Text1.Text);
            mem = Convert.ToInt32(f.Length);

            if ( Check1.Checked == true )
            {
                sp = 1;
            }
            else
            {
                sp = 0;
            }

            if ( Check2.Checked == true )
            {
                po = 1;
            }
            else
            {
                po = 0;
            }

            if ( Check3.Checked == true )
            {
                fa = 1;
            }
            else
            {
                fa = 0;
            }

            if ( Check4.Checked == true )
            {
                cl = 1;
            }
            else
            {
                cl = 0;
            }

            if ( Check5.Checked == true )
            {
                bl = 1;
            }
            else
            {
                bl = 0;
            }

            if ( Check6.Checked == true )
            {
                hv = 1;
            }
            else
            {
                hv = 0;
            }

            if ( Check7.Checked == true )
            {
                wl = 1;
            }
            else
            {
                wl = 0;
            }

            pa = Convert.ToInt16(Text5.Text);
            st = Convert.ToInt16(Text6.Text);

                  sPfad = uni_enc.GetBytes(Text1.Text);
                  tPfad = uni_enc.GetBytes(Text3.Text);
                  title = uni_enc.GetBytes(Text4.Text);

                  tmpMemPath = Marshal.AllocHGlobal(mem);
            tmpMemPath = (IntPtr)Module1.GetPDFText(sPfad, 3, hv, fa,
tPfad, sp, title, po, pa, cl, bl, st, wl);
            Text2.Text = Marshal.PtrToStringUni(tmpMemPath);
                  tmpMemPath = IntPtr.Zero;

            }

`...
```

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,"",1,0,0
            ,0,0,0);
            FreeLibrary(m_hInstLib);
        }
}
. . .
```