3-Heights™ PDF Creator Library

Version 4.5

User Manual

Contact: pdfsupport@pdf-tools.com

Owner: **PDF Tools AG**

Kasernenstrasse 1
8184 Bachenbülach
Switzerland

http://www.pdf-tools.com

Table of Contents

1 Introduction

1.1 Description

The 3-Heights™ PDF Creator Library is a component to create PDF/A-1 compliant documents. The interface is based on a comprehensible object model, which reflects the functional range of PDF/A-1. The interface is designed in a way that it can be easily enhanced. It will continuously be adapted to new requirements, such as the evolvement of the PDF/A standard. The component ensures that all PDF/A provisions are enforced, such as file formatting rules, the embedding of fonts and color profiles, and many more.

1.2 Functions

Features

There is a C# sample project called "PDFCreatorSample" that creates the file "PDFCreatorAPINet_test.pdf". This PDF file shows many of the features. Here is a summary:

- Path
  - Single and Multi-segment lines
  - Rectangle, Circle, Bezier curves
  - Filling, Stroking, Clipping and combinations thereof
  - Line width, cap, join, dash array, dash phase and miter limit
- Text
  - Font size, Character spacing, Word spacing
  - Horizontal scaling, Leading, Rise
  - Standard Type 1 fonts, TrueType fonts
  - Unicode characters
  - Text stroke line width, line join and dashes
  - Fill and stroke text, invisible text
  - Use text as clipping path
  - RGB and CMYK colors
- Images
  - Bi-level: CCITT G3, G3 2D and G4, Flate, LZW, Packbits, uncompressed
  - 4 bit and 8 bit grayscale: Flate, LZW, Packbits, JPEG and JPEG-6 (8 bit only), uncompressed
  - RGB: Flate, JPEG and JPEG-6, LZW, Packbits, uncompressed
- Transformations

  - Translation
  - Scaling
  - Skewing (Horizontal, Vertical)
  - Rotation
- Metadata
  - Document information entries: Title, Author, Subject, Keywords, Creator, (Producer), Custom entries

Compliance

- Standards: ISO 19005-1 (PDF/A-1), ISO 19005-2 (PDF/A-2), ISO 19005-3 (PDF/A-3), ISO 32000 (PDF 1.7), TIFF V6
- Quality assurance: Isartor test suite

1.3 Interfaces

The following interfaces are available:

- C
- Java
- .NET
- COM

1.4 Operating Systems

- Windows XP, Vista, 7, 8, 8.1 – 32 and 64 bit
- Windows Server 2003, 2008, 2008-R2, 2012, 2012-R2 – 32 and 64 bit
- HP-UX 11 and later PA-RISC2.0 32 bit or HP-UX 11i and later ia64 (Itanium) 64 bit
- IBM AIX 5.1 and later (64 bit)
- Linux (32 and 64 bit)
- Mac OS X 10.4 and later (32 and 64 bit)
- Sun Solaris 2.8 and later – SPARC and Intel
- FreeBSD 4.7 and later 32 bit or FreeBSD 9.3 and later 64 bit (on request)

2 Installation

2.1 Interface specific Installation Steps

COM Interface

Registration: Before you can use the 3-Heights™ PDF Creator Library component in your COM application program you have to register the component using the *regsvr32.exe* program that is provided with the Windows operating system. The following command shows the registration of *PdfCreatorAPI.dll*.

```
C:\>regsvr32.exe PdfCreatorAPI.dll
```

The registration can also be done silently (e.g. for deployment) using the switch /s.

Other Files: The other DLLs do not need to be registered, but for simplicity it is suggested that they are in the same directory as the *PdfCreatorAPI.dll*.

Java Native Interface

For compilation and execution: The Java Archive *jar\plba.jar* needs to be on the class search path. This can be done by either adding it to the environment variable CLASSPATH, or by specifying it using the switch –classpath.

```
javac -classpath .;C:\pdf-tools\jar\plba.jar sample.java
```

For execution: Additionally the Library *bin\PdfCreatorAPI.dll* needs to be on the library path. This can be achieved by either adding it to the environment variable PATH, or by specifying it using the switch -Djava.library.path.

```
java -classpath .;C:\pdf-tools\jar\plba.jar -Djava.library.path=.;C:\pdf-
tools\bin sample
```

.NET Interface

The 3-Heights™ PDF Creator Library does not provide a pure .NET solution. Instead, it consists of .NET assemblies, which are added to the project and a native DLL which is called by the .NET assemblies. This has to be accounted for when installing and deploying the tool.

The .NET assemblies (**NET.dll*) are to be added as references to the project. They are required at compilation time.

PdfCreatorAPI.dll is not a .NET assembly, but a native DLL. It is not to be added as a reference in the project.

The native DLL *PdfCreatorAPI.dll* is called by the .NET assembly *PdfCreatorNET.dll*. *PdfCreatorAPI.dll* must be found at execution time by the Windows operating system.

There are various approaches to ensure this. Below two are listed:

- *PdfCreatorAPI.dll* is copied to a directory that is on the environment variable "PATH". e.g. it is either copied to
    - a new dedicated directory, which is then added to the "PATH", or

  - an existing directory which already is on the "PATH", such as *%SystemRoot%\System32* (e.g. *C:\Windows\System32*).

  This approach is usually used for development.

- *PdfCreatorAPI.dll* is copied to the directory where the executable resides.

  If you look at the C# example that is provided by PDF Tools: This is the same directory as where *PDFCreatorSample.exe* resides, which is something like *PDFCreatorSample\bin\Debug* (or equivalent for release).

  This approach is usually used for deployment.

C Interface

- The header file *pdfcreatorapi_c* needs to be included in the C/C++ program.
- The Object File Library *PdfCreatorAPI.lib* needs to be linked to the project.
- *PdfCreatorAPI.dll* should be on the environment variable PATH or, if using MS Visual Studio, in the directory for executable files.

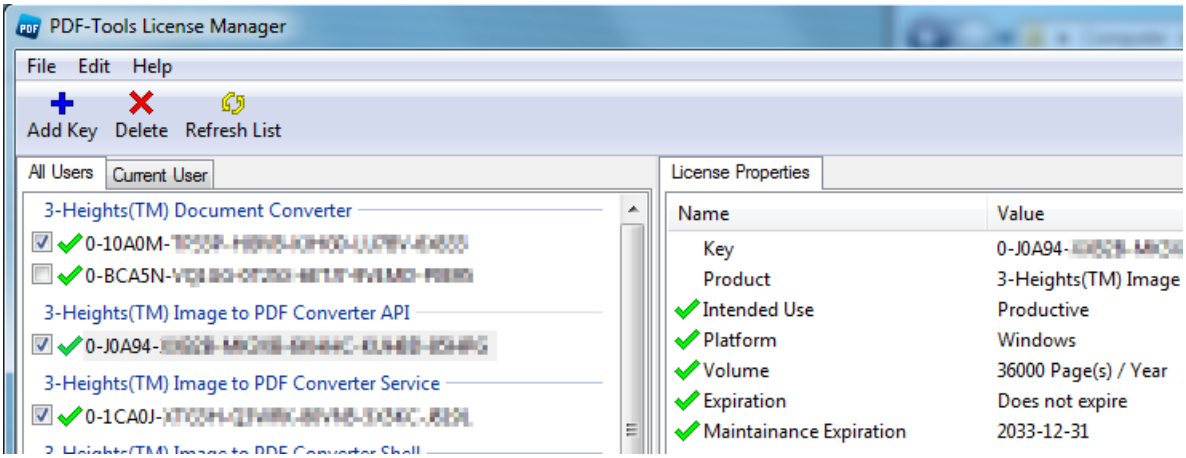
3 License Management

There are three possibilities to pass the license key to the application:

1. The license key is installed using the GUI tool (Graphical user interface). This is the easiest way if the licenses are managed manually. It is only available on Windows.
2. The license key is installed using the shell tool. This is the preferred solution for all non-Windows systems and for automated license management.
3. The license key is passed to the application at runtime via the "LicenseKey" property. This is the preferred solution for OEM scenarios.

3.1 Graphical License Manager Tool

The GUI tool *LicenseManager.exe* is located in the *bin* directory of the product kit.



List all installed license keys

The license manager always shows a list of all installed license keys in the left pane of the window. This includes licenses of other PDF Tools products.

The user can choose between:

- Licenses available for all users. Administrator rights are needed for modifications.
- Licenses available for the current user only.

Add and delete license keys

License keys can be added or deleted with the "Add Key" and "Delete" buttons in the toolbar.

- The "Add key" button installs the license key into the currently selected list.
- The "Delete" button deletes the currently selected license keys.

Display the properties of a license

If a license is selected in the license list, its properties are displayed in the right pane of the window.

Select between different license keys for a single product

More than one license key can be installed for a specific product. The checkbox on the left side in the license list marks the currently active license key.

3.2 Command Line License Manager Tool

The command line license manager tool *licmgr* is available in the *bin* directory for all platforms except Windows.

A complete description of all commands and options can be obtained by running the program without parameters:

```
licmgr
```

List all installed license keys

```
licmgr list
```

The currently active license for a specific product ist marked with a star '*' on the left side.

Add and delete license keys

Install new license key

```
licmgr store X-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

Delete old license key

```
licmgr delete X-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

Both commands have the optional argument `-s` that defines the scope of the action:

- `g`: For all users
- `u`: Current user

Select between different license keys for a single product

```
licmgr select X-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

3.3 License Key Storage

Depending on the platform the license management system uses different stores for the license keys.

Windows

The license keys are stored in the registry:

- HKLM\Software\PDF Tools AG (for all users)

- HKCU\Software\PDF Tools AG (for the current user)

Mac OS X

The license keys are stored in the file system:

- /Library/Application Support/PDF Tools AG (for all users)
- ~/Library/Application Support/PDF Tools AG (for the current user)

Unix / Linux

The license keys are stored in the file system:

- /etc/opt/pdf-tools (for all users)
- ~/.pdf-tools (for the current user)

Note: The user, group and permissions of those directories are set explicitly by the license manager tool.

It may be necessary to change permissions to make the licenses readable for all users. Example:

```
chmod -R go+rx /etc/opt/pdf-tools
```

4 User's Guide

C# sample project called "PDFCreatorSample" is a very good basis to start with, it shows virtually all features provided in the library.

4.1 Fonts

Fonts can be set in different ways. When using the function *PdfNewInstalledFont* a font is set using its family name and its style. If no style is defined, the base font is used.

Example:

The following two samples (in C++) set the font Arial and Arial-Bold

```
TPdfFont* pFont = PdfNewInstalledFont(pDocument, "Arial", NULL, 1);
TPdfFont* pFont = PdfNewInstalledFont(pDocument, "Arial", "Bold", 1);
```

For 12 out of the 14 PDF Standard fonts there is a special behavior: Fonts are replaced if they are not found on the local system. The replacement is as shown in the table below:

Family Name	Style	Replacement Font
Courier		CourierNew
Courier	Oblique	*CourierNew,Italic*
Courier	Bold	**CourierNew,Bold**
Courier	BoldOblique	***CourierNew,BoldItalic***

Helvetica		Arial
Helvetica	Oblique	*Arial,Italic*
Helvetica	Bold	**Arial,Bold**
Helvetica	BoldOblique	***Arial,BoldItalic***
Times-Roman		TimesNewRoman
Times	Italic	*TimesNewRoman,Italic*
Times	Bold	**TimesNewRoman,Bold**
Times	BoldItalic	***TimesNewRoman,BoldItalic***

The two PDF Standard Fonts "Symbol" and "ZapfDingbats" are not replaced.

Example:

If the local system has Helvetica and Helvetica-Bold installed, they will be used, otherwise Arial and Arial-Bold will be selected as replacement font.

```
TPdfFont* pFont = PdfNewInstalledFont(pDocument, "Helvetica", NULL, 1);
TPdfFont* pFont = PdfNewInstalledFont(pDocument, "Helvetica", "Bold", 1);
```

5 Examples

This section provides a selection of small C# samples which cover one specific topic each.

5.1 Text

```
private void buttonPlain_Click(object sender, EventArgs e)
{
  // Create document and set its document attributes
  string fileName = "C:\\temp\\text.pdf";
  Document document1 = PdfCreatorAPI.GetInstance().NewDocument(fileName,
Conformance.ePdfA1b);

  // Create a page
  Page page1 = document1.NewPage(new PdfTools.Pdf.Rectangle(0, 0, 595, 421));

  // Create a text element (requires a graphics and font object)
  PdfTools.Pdf.Graphics g1 = document1.NewGraphics();
  PdfTools.Pdf.Font font1 = document1.NewFont("Helvetica", "BoldOblique");
  PdfTools.Pdf.Text text1 = g1.NewText(font1, 20, "Hello World", 100, 300);
  g1.FillText(text1, null);

  // Add graphics to page
  page1.AddGraphics(g1);

  // Add page to document and close
  document1.AddPage(page1);
  document1.Close();
}
```

5.2 Image

```
private void buttonImage_Click(object sender, EventArgs e)
{
  string fileName = "C:\\temp\\image.pdf";
  Document document1 = PdfCreatorAPI.GetInstance().NewDocument(fileName,
Conformance.ePdfA1b);

  Page page1 = document1.NewPage(new PdfTools.Pdf.Rectangle(0, 0, 595, 421));

  // Create an image element (requires a graphics object)
  PdfTools.Pdf.Graphics g1 = document1.NewGraphics();
  PdfTools.Pdf.Image image1 = document1.NewImage("input.gif");
  // Save and later restore the graphics state to ensure the transformation
  // has no sideeffects
  g1.Save();
  g1.ModifyTransform(g1.NewTransform(image1.Width, 0, 0, image1.Height, 10, 10));
  g1.PaintImage(image1);
  g1.Restore();

  page1.AddGraphics(g1);
  document1.AddPage(page1);
  document1.Close();
}
```

5.3 Vector Graphic

```
private void buttonVector_Click(object sender, EventArgs e)
{
  string fileName = "C:\\temp\\vector.pdf";
  Document document1 = PdfCreatorAPI.GetInstance().NewDocument(fileName,
Conformance.ePdfA1b);

  Page page1 = document1.NewPage(new PdfTools.Pdf.Rectangle(0, 0, 595, 421));

  PdfTools.Pdf.Graphics g1 = document1.NewGraphics();

  // Add two Bezier curves to approximate a circle
  {
    double x = 100;
    double y = 117;
    double s = 100;
    double k = 0.66; // guessed apporoximation
    Path path1 = g1.NewPath();
    path1.MoveTo(x, y);
    path1.BezierTo(x, y + s * k, x + s, y + s * k, x + s, y);
    path1.BezierTo(x + s, y - s * k, x, y - s * k, x, y);
    Paint paint1 = g1.NewRGBPaint(216 + 256 * 216 + 256 * 256 * 216);
    g1.FillPath(path1, paint1);
  }

  // Add rectangles with different colors
  for (int row = 0; row <= 5; row++)
  {
    for (int col = 0; col <= 5; col++)
    {
      Paint paint2 = g1.NewRGBPaint((uint)(256 * 256 * 255 * (1 - col * 0.2) +
                     256 * 255 * (row * 0.2) + 255 * (col - row) * 0.2));
      Path path2 = g1.NewPath();
      path2.AddRectangle(100 + col * 11, 100 + row * 11, 9, 9);
```

```
      g1.FillPath(path2, paint2);
    }
  }

  page1.AddGraphics(g1);
  document1.AddPage(page1);
  document1.Close();
}
```

5.4 Clipping

```
private void buttonClip_Click(object sender, EventArgs e)
{
  string fileName = "C:\\temp\\clip.pdf";
  Document document1 = PdfCreatorAPI.GetInstance().NewDocument(fileName,
Conformance.ePdfA1b);

  Page page1 = document1.NewPage(new PdfTools.Pdf.Rectangle(0, 0, 595, 421));

  // Create an image element (requires a graphics object
  PdfTools.Pdf.Graphics g1 = document1.NewGraphics();
  PdfTools.Pdf.Image image1 = document1.NewImage("input.gif");

  // Set clipping path
  Path path1 = g1.NewPath();
  path1.AddCircle(110, 130, 90);
  g1.IntersectClipPath(path1);

  g1.Save();
  g1.ModifyTransform(g1.NewTransform(image1.Width, 0, 0, image1.Height, 10, 10));
  g1.PaintImage(image1);
  g1.Restore();

  page1.AddGraphics(g1);
  document1.AddPage(page1);
  document1.Close();
}
```