User Manual

3-Heights™ PDF Viewer API

Version 4.5

PDF/A compliant

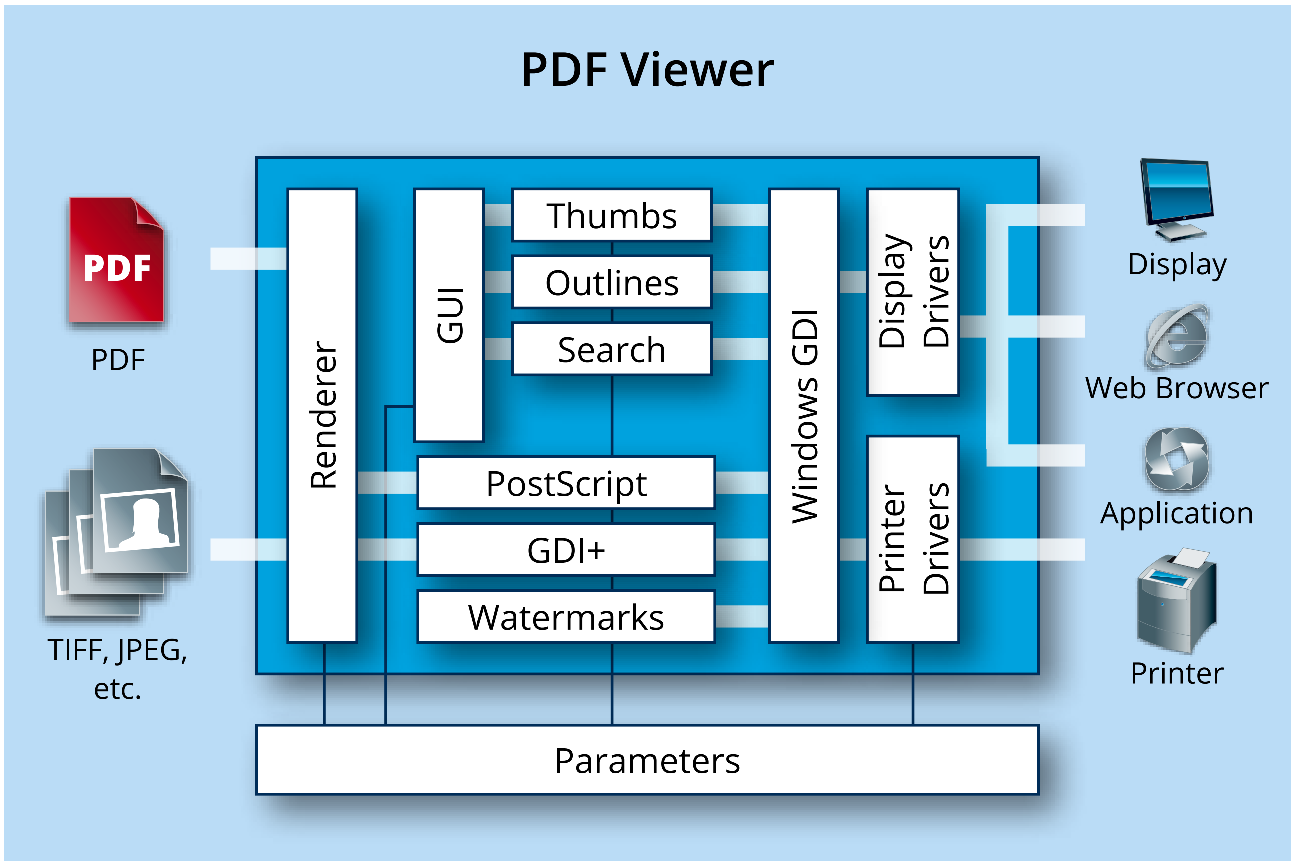
PDF-TOOLS.COM
Premium PDF Technology

Contents

1 Introduction

1.1 Description

The 3-Heights™ PDF Viewer API is an independent, compact, high-quality PDF viewer. It offers a multitude of functions such as display options, navigation, optional printing and support for non-PDF formats such as TIFF, JPEG and other raster image formats. The PDF Viewer is characterized mainly by its ease of integration and handling and high performance.



The Viewer can be used as an intranet website tool, for instance, or as a standalone application or software solution component. The Viewer can be started outside of the programming environment in under a second via the command line, and integrates seamlessly in modern development environments as an API.

1.2 Functions

The 3-Heights™ PDF Viewer API offers many options for displaying PDF documents in a manner that suits requirements: files can be viewed in single page or multi-page mode, and the navigation supports links, bookmarks and calling up pages arbitrarily. The search function can locate strings of text in the document. The tool supports PDF documents in many languages: the reproduction of Japanese, Chinese, Korean, Russian, etc., is no problem at all. The PDF Viewer can also print out and save PDF documents.

Viewing

- Display PDF file in single page or multi-page mode
- Enter password to decrypt PDF documents
- Display image files such as TIFF, JPEG, JBIG2, JPEG2000, PNG and GIF
- Open files from file system or from internet (HTTP, HTTPS, FTP)
- Query the number of pages in a document
- Set scalability (page size, page width, actual size, any size)

- Set X,Y resolution in dpi
- Set various display options such as image filters
- Reproduce documents with Chinese, Japanese and Korean fonts (CJK)
- Read document from file or memory
- Convert viewer coordinates to PDF coordinates
- Set number of pages per window
- Rotate and display the page
- Show annotations
- Support for "FDF" file format
- Save the PDF or image files on display
- Set user rights for the saved new version of the PDF file
- Compare two documents and highlight differences
- Display, open and save embedded files, e.g. of PDF Portfolios

Navigation

- Search PDF documents for a specific string of text
- Show and hide windows for bookmarks and page navigation
- Jump to a bookmark's location
- Jump to the location of a link within the document
- Move windows vertically and horizontally using the mouse or keyboard
- Enlarge and reduce windows with the aid of a zoom rectangle or zoom factor
- Freely select any page in the document for display
- Query the position of the cursor
- Query the position of the scrollbars
- Intercept events (e.g. mouse events, keyboard events, zoom events)
- Highlight rectangle in color
- Set mouse cursor mode
- Show and hide scrollbars
- Activate or deactivate links
- Activate mouse wheel (for scrolling) or deactivate
- Replace embedded fonts with preinstalled fonts for viewing and printing
- Display fonts as vector graphics
- Navigate within embedded files, e.g. of PDF Portfolios

Printing

- Print the PDF or image files on display to any local or remote Windows printer
- RAW or EMF mode for printing
- Duplex printing
- Set paper tray
- Windows 9x compatibility mode to support legacy printers
- Set paper size
- Set number of copies
- Centered printing
- Set print orientation (landscape, portrait, printer default, automatic)
- Query local Windows standard printer
- Optional suppression of digital signatures during printing
- Option to pre-render pages and send them to the printer as bitmaps
- Black point compensation during printing

Annotate Document

- Create, modify and delete annotations on PDF documents.
- Enable and disable possibility to edit annotations.
- Support Sticky Notes, Text and Highlight Annotations.
- Configure default values for properties of new annotations that cannot be overridden by the user.
- Load annotations from and store annotations to FDF file.
- Save files on the file system or on the internet (WebDAV)

Formats

Input Formats

- PDF 1.0 - 1.7, PDF/A
- Raster images (e.g. TIFF, JPEG, PNG, GIF, BMP)

Output Formats

- The PDF Viewer can save files in their original input format.
- Forms Data Format (FDF)

Compliance

Standards: ISO 32000-1 (PDF 1.7), ISO 19005-1 (PDF/A-1), ISO 19005-2 (PDF/A-2)

1.3 Interfaces

There is a C and a COM interface available.

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

1.5 Compatibility Note

1. In versions prior to 2010 it was distinguished between the standard and the pro version. Certain features (such as printing or viewing raster images) were only supported in the pro version. At present there is only one version, which supports all features.
2. The Viewer control is an ActiveX control. It is provided in two different versions: as 32 and 64 bit. Since it runs in-process, the 64 bit version can only run with a 64 bit application. This means for development with MS Studio, which is a 32 bit application, the 32 bit version of the Viewer control must be used. For deployment however, the 64 bit version can be used.

2 Installation

2.1 Windows

The retail version of the 3-Heights™ PDF Viewer API comes as a ZIP archive containing various files including runtime binary executable code, files required for the developer, documentation and license terms.

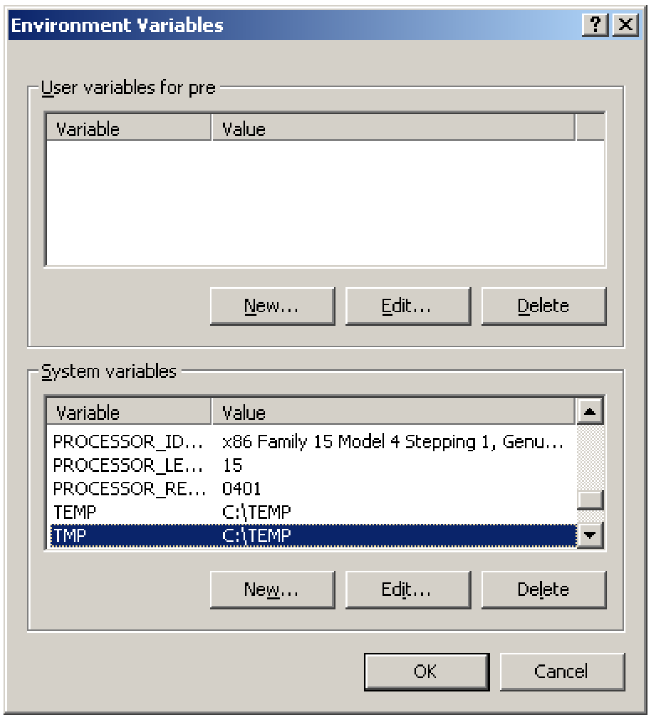
1. Download the ZIP archive of the product from your download account at http://www.pdf-tools.com.
2. Unzip the file using a tool like WinZip available from WinZip Computing, Inc. at http://www.winzip.com to a directory on your hard disk where your program files reside (e.g. *C:\Program Files\PDF Tools AG*).
3. Check the appropriate option to preserve file paths (folder names). The unzip process now creates the following subdirectories:

bin:	Contains the runtime executable binary code.
bin\fonts:	Contains the PDF standard fonts and the font mapping file. Copy and thereby install the fonts to the OS fonts directory (*%systemroot%\fonts*, e.g. *C:\Windows\fonts*).
bin\icc:	Contains the two color profiles "USWebCoatedSWOP.icc" and "sRGB Color Space Profile.icm".
doc:	Contains documentation files.
include:	Contains header files to include in your C / C++ project.
samples:	Contains sample programs in various programming languages.

   There is the option to download the software as MSI file, which makes the installation easier.
4. Optionally register your license key using the License Manager.
5. Identify which interface you are using. Perform the specific installation steps for that interface described in chapter Interfaces.

Ensure the two system environment variables TEMP and TMP exist and point to an existing directory. This directory is required to temporarily install fonts that are embedded in PDF documents.

Control Panel ->System ->Advanced ->Environment Variables



Variable	Value
PROCESSOR_ID...	x86 Family 15 Model 4 Stepping 1, Genu...
PROCESSOR_LE...	15
PROCESSOR_RE...	0401
TEMP	C:\TEMP
TMP	C:\TEMP

General

Here is an overview of the relevant files that come with the 3-Heights™ PDF Viewer API Tool:

`bin\viewerOCX.dll`	This is the DLL that contains the main functionality (required).
`bin\pdcjk.dll`	This DLL contains support for Asian languages (optional). It is loaded from the module path.
`bin\inet.dll`	This DLL implements http:, https: and ftp: connections using the Internet Explorer (optional). It is loaded from module path.
`bin\Fonts\*.ttf`	This is the TrueType version of the font ZapfDingbats.
`bin\Fonts\fonts.ini`	This configuration file allows for specifying substitution fonts (optional).

2.2 Uninstall, Install a New Version

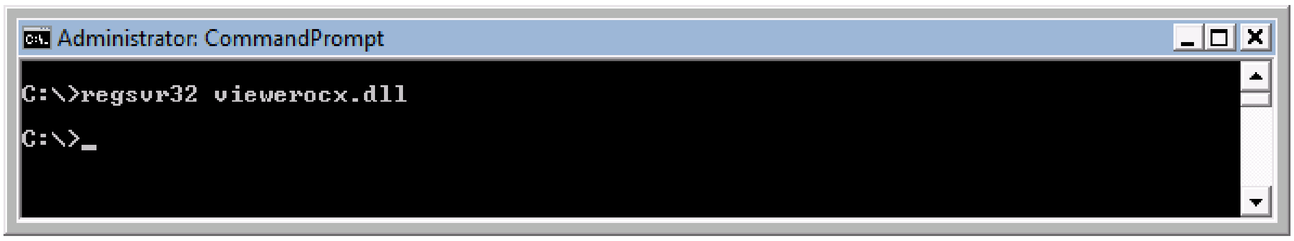
If you used the MSI for the installation, go to Start ->3-Heights™ PDF Viewer API ... ->Uninstall...
If you used the ZIP file: In order to uninstall the product undo all the steps done during installation, e.g. un-register using regsvr32 -u, delete all files, etc.

Installing a new version does not require to previously uninstall the old version. The files of the old version can directly be overwritten with the new version. If using the COM interface, the new DLL must be registered, un-registering the old version is not required.

2.3 Interfaces

COM Interface

Before you can use the 3-Heights™ PDF Viewer API 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; it resides in the directory System32. If you are using a newer operating system, such as Vista or Windows 7, start the command prompt as Administrator. The following screenshot shows the registration of the PDF Viewer API DLL.

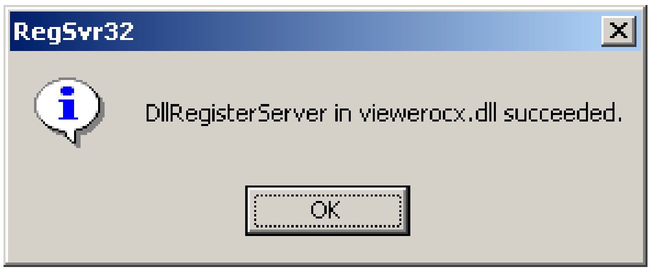


```
C:\>regsvr32 viewerocx.dll

C:\>_
```

If you are using a 64-bit operating system and would like to register the 32-bit version of the 3-Heights™ PDF Viewer API, you need to use the regsvr32 from the directory `%SystemRoot%\SysWOW64` instead of `%SystemRoot%\System32`.[1]

If the registration process succeeds, the following dialog window is displayed:



DllRegisterServer in viewerocx.dll succeeded.

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

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

[1] Otherwise you get the following message: *"LoadLibrary("viewerOCXAPI.dll") failed - The specified module could not be found."*

C Interface

- The header file *viewer_c.h* needs to be included in the C/C++ program.
- The library *lib\viewerOCX.lib* needs to be linked to the project.
- The dynamic link library *bin/viewerOCX.dll* needs to be in path of executables (e.g. on the environment variable "PATH").

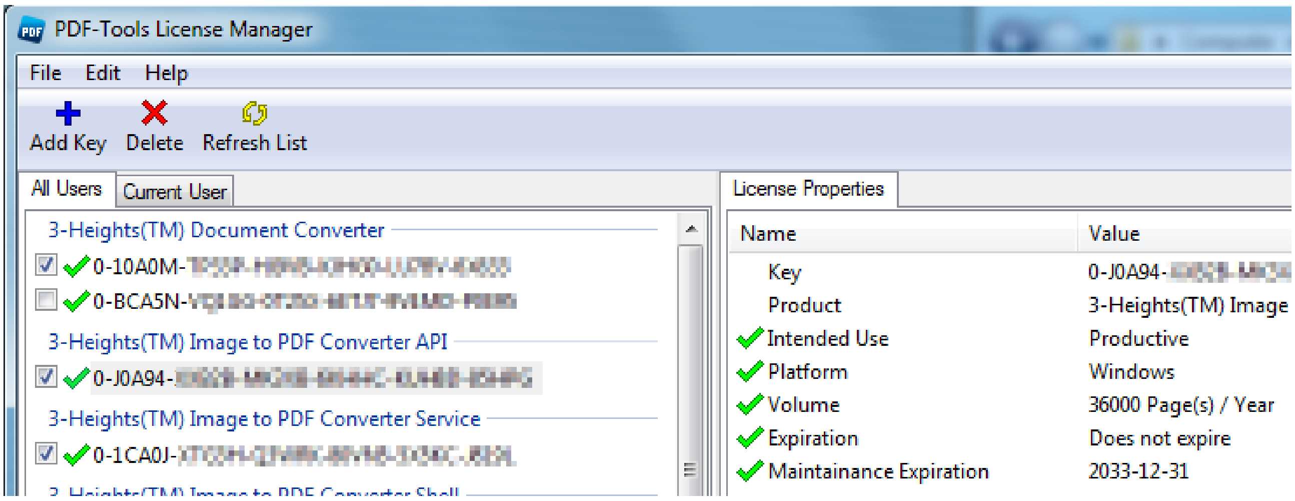
3 License Management

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.



3-Heights(TM) Document Converter
- [✓] 0-10A0M-[redacted]
- [ ] 0-BCA5N-[redacted]

3-Heights(TM) Image to PDF Converter API
- [✓] 0-J0A94-[redacted]

3-Heights(TM) Image to PDF Converter Service
- [✓] 0-1CA0J-[redacted]

3-Heights(TM) Image to PDF Converter Shell

Name	Value
Key	0-J0A94-[redacted]
Product	3-Heights(TM) Image
✓ Intended Use	Productive
✓ Platform	Windows
✓ Volume	36000 Page(s) / Year
✓ Expiration	Does not expire
✓ Maintainance Expiration	2033-12-31

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
```

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:

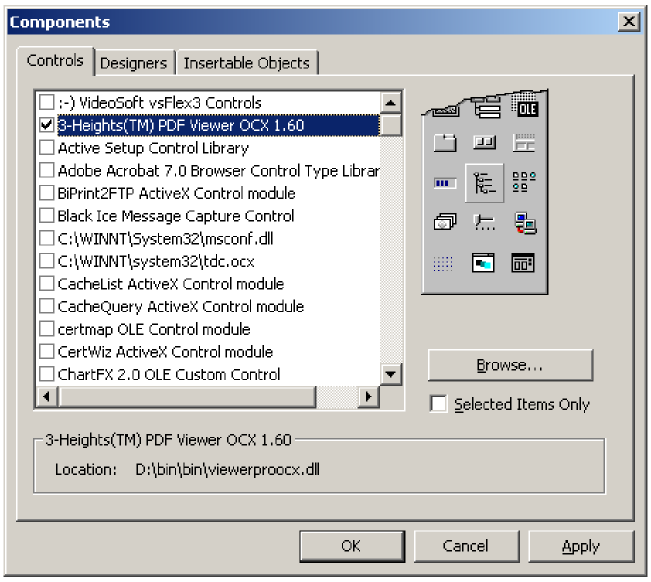
- `HKLM\Software\PDF Tools AG` (for all users)
- `HKCU\Software\PDF Tools AG` (for the current user)

4 Programming Interfaces

4.1 Visual Basic

After installing and registering the PDF Viewer, you find a Visual Basic example `PDFViewer.vbp` in the directory `samples\VB\`. You can either use this sample as a basis for an application, or you can start from scratch.

If you start from scratch, first create a new Standard-Exe Visual Basic 6 project. Add the 3-Heights™ PDF Viewer API OCX component:



Components

Controls | Designers | Insertable Objects

- [ ] :-) VideoSoft vsFlex3 Controls
- [x] 3-Heights(TM) PDF Viewer OCX 1.60
- [ ] Active Setup Control Library
- [ ] Adobe Acrobat 7.0 Browser Control Type Librar
- [ ] BiPrint2FTP ActiveX Control module
- [ ] Black Ice Message Capture Control
- [ ] C:\WINNT\System32\msconf.dll
- [ ] C:\WINNT\system32\tdc.ocx
- [ ] CacheList ActiveX Control module
- [ ] CacheQuery ActiveX Control module
- [ ] certmap OLE Control module
- [ ] CertWiz ActiveX Control module
- [ ] ChartFX 2.0 OLE Custom Control

Browse...

- [ ] Selected Items Only

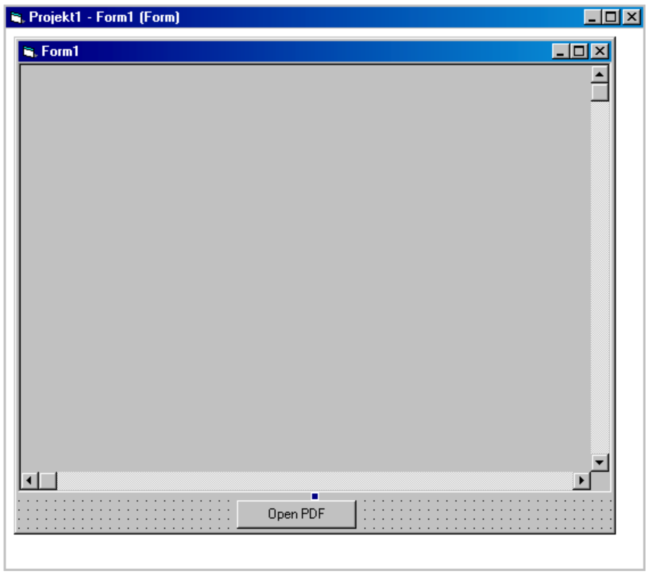
3-Heights(TM) PDF Viewer OCX 1.60

Location: D:\bin\bin\viewerproocx.dll

OK | Cancel | Apply

This will add the PDF Viewer API object icon, which looks like this:

Add PDF Viewer object and a command button, so that you receive a form that looks similar as the one shown below:



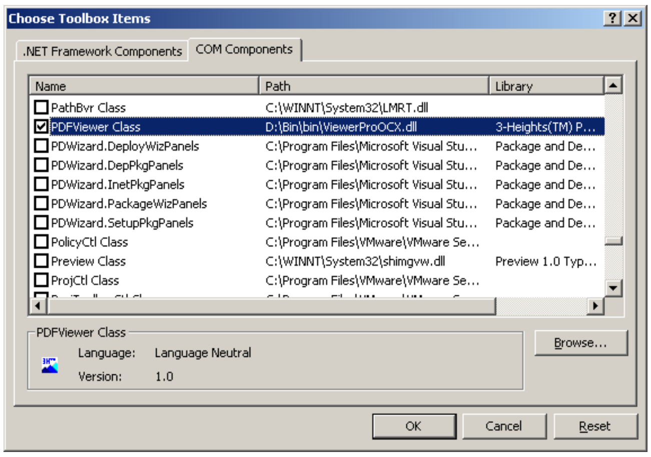
Now double-click the command button, add an open function and your PDF Viewer API is already completed. The open function is the first function described in the next chapter "User's Guide".

4.2 .NET

The 3-Heights™ PDF Viewer API does not provide a native .NET interface. In a .NET project the component is used through its COM interface.

Therefore the component first needs to be registered as described in the chapter “COM Interface”.

When using the component in a new project, add the PDFViewer Class to the Toolbox. To do so, right-click on the expanded MS Studio’s Toolbox and select “Choose Items...>”. In the dialog select the COM component “PDF Viewer Class” as shown in the screenshot below:



Choose Toolbox Items

.NET Framework Components | COM Components

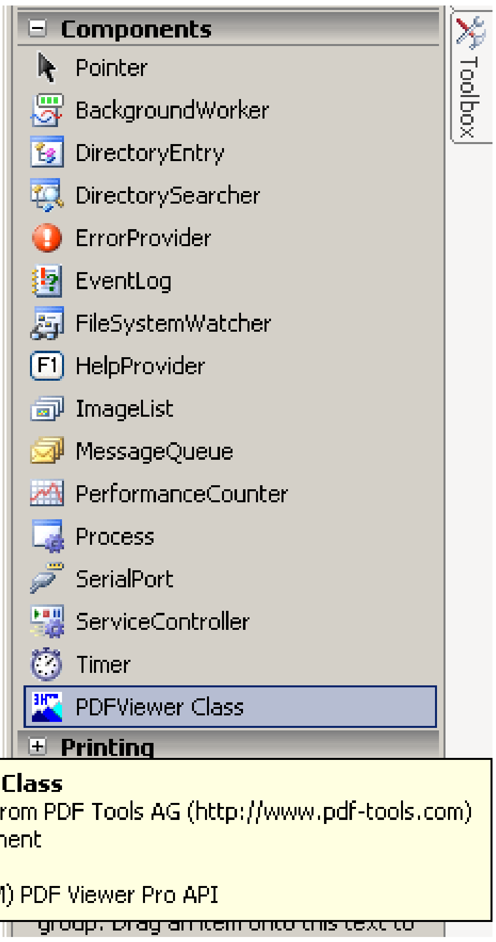
Name	Path	Library
☐ PathBvr Class	C:\WINNT\System32\LMRT.dll	
☑ PDFViewer Class	D:\Bin\bin\ViewerProOCX.dll	3-Heights(TM) P...
☐ PDWizard.DeployWizPanels	C:\Program Files\Microsoft Visual Stu...	Package and De...
☐ PDWizard.DepPkgPanels	C:\Program Files\Microsoft Visual Stu...	Package and De...
☐ PDWizard.InetPkgPanels	C:\Program Files\Microsoft Visual Stu...	Package and De...
☐ PDWizard.PackageWizPanels	C:\Program Files\Microsoft Visual Stu...	Package and De...
☐ PDWizard.SetupPkgPanels	C:\Program Files\Microsoft Visual Stu...	Package and De...
☐ PolicyCtl Class	C:\Program Files\VMware\VMware Se...	
☐ Preview Class	C:\WINNT\System32\shimgvw.dll	Preview 1.0 Typ...
☐ ProjCtl Class	C:\Program Files\VMware\VMware Se...	

PDFViewer Class

Language: Language Neutral

Version: 1.0

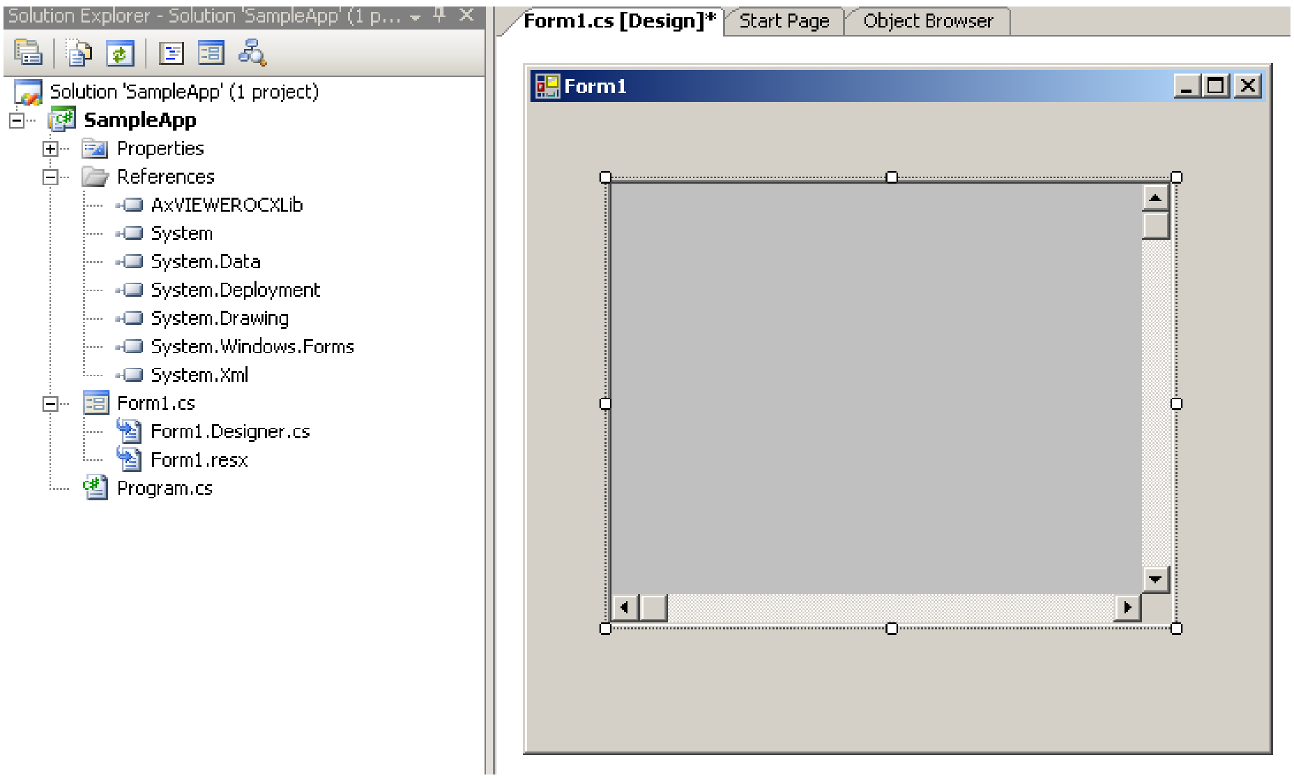
After this step the PDFViewer Class is available in MS Studio’s Toolbox as indicated in the screenshot below:



PDFViewer Class
Version 1.0 from PDF Tools AG (http://www.pdf-tools.com)
COM Component

3-Heights(TM) PDF Viewer Pro API

Use this class to draw a PDFViewer component onto a form. This step creates the COM-Interop DLL `Interop.VIEWEROCXLib.dll` and refers to it as `AxVIEWEROCXLib` in the project. The screenshot below shows this.



The `Interop.VIEWEROCXLib.dll` is automatically generated when launching an MS Studio project that uses the `PDFViewer` Class and `ViewerOCX.dll` is registered on that OS.

4.3 MFC

Please refer to the sample "mfcviewer", which is created using the MFC wizard and is provided with the software package.

5 User's Guide

Most samples in this guide are written in Visual Basic. The call sequence, however, for any other programming language is the same.

5.1 Open a PDF Document

Documents can be opened either from file using the method Open or from memory using the method Open-Mem. A password is required and must be provided if the file is encrypted with a user password. For documents that have no user password set, use the default password - an empty string. Here is a simple VB example that shows how to open a PDF file from disk.

Example: Open PDF from File System

```
Private Sub Open_Click()
  If Not ViewerControl.Open(App.Path & "\input.pdf", "") Then
     MsgBox "Failed to open input file"
  End If
End Sub
```

ViewerControl is an object of type PDFViewer. Its name is defined by the name of the viewer control. The sample checks whether opening the input file was successful or not. If not, it shows a message box.

OpenMem is usually used when a PDF document is already available in memory, e.g. is read from a data base or is passed in-memory from another application. The following example shows how a PDF document can be opened form memory by reading it from file and writing it in a byte array and then reading that byte array using OpenMem.

Example: Open PDF from Memory

```
Private Sub OpenMem_Click()
  Dim bChar() As Byte
  Dim lFileLenght As Long
  Open App.Path & "\input.pdf" For Binary As #1
    lFileLenght = LOF(1)
    ReDim bChar(lFileLenght - 1)
    Get #1, , bChar
  Close #1
  ViewerControl.OpenMem bChar, ""
End Sub
```

5.2 Navigate

There are various ways how the user can navigate in a document with the viewer control. The application programmer has all options to enable, disable or limit the user's navigation. The following features can be used to navigate:

- Use the property Page to set page number to be displayed. The page range goes from 1 to PageCount.
- Use the mouse in cursor mode eCursorModeMove (default) to scroll the page by pressing the left mouse button and moving the mouse up and down. Change the curser mode using the property CurserMode.
- Use horizontal and vertical scroll bars. Scroll bar functionality can be disabled using the property ViewerOptions and disable (AND NOT[2]) eViewerOptionScrollbars.
- Use the mouse wheel to scroll. The mouse wheel can be disabled using the property ViewerOptions and enable (OR) eViewerOptionDisableMouseWheel.
- Use the method Goto to move to a specific location and zoom level within the document.
- Use the outlines (bookmarks) or thumbnails from the navigation panel at left hand side of the control to random access a position.
  - The navigation panel can be disabled using the property ViewerOptions and disable eViewerOptionPages and eViewerOptionOutlines from it.
  - Enable eViewerOptionPages, disable eViewerOptionOutlines to show thumbnails
  - Enable eViewerOptionOutlines, disable eViewerOptionPages to show outlines; if there are not outlines in the document, the navigation panel is not displayed.
- Internal and external links within the content of the PDF document can also be used to navigate.
  - Internal links (Goto) are disabled using eViewerOptionDisableGoto.
  - External links (GotoR) are disabled using eViewerOptionDisableGotoR.
  - URI are disabled using eViewerOptionDisableURI.
- The search functions can indirectly be used to navigate.

[2] Options are enabled using bitwise OR and disabled using bitwise AND NOT. See also chapter TPDFRendererOption.

5.3 Start a Print Job

A print job can be started using BeginDocument. The parameter of BeginDocument is the name of the print job. To define the end the print job, use EndDocument. The method PrintFile starts a print job, prints, and ends the print job. It cannot be combined with other calls, such as BeginDocument or EndDocument.

What Is a Print Job?

A print job is a series of pages that are printed as one job. All pages of a print job are printed before the next print job starts. The order in which individual print jobs are processed by the printer device is defined by the spooler. The order in which print jobs are created doesn't have to be the same order as they are printed. Small print jobs may receive higher priority and can overtake large print jobs. An exception to this are linked print jobs.

Individual Pages

The 3-Heights™ PDF Viewer API can print the document on display or a page thereof. It is not intended to be used for automated bulk printing.

The following sample shows how to open a printer, how to start and name a print job and how to print two copies. It assumes a document is already opened and displayed.

Example:

```
Private Sub Print_Click()
  ' replace the printer name with your printer name
  ViewerControl.OpenPrinter "HP LaserJet 4050 Series PS"
  ' start a new print job called "MyJob"
  ViewerControl.BeginDocument "MyJob"
  ' loop through the number of copies
  NumberOfCopies = 2
  For iCopies = 1 To NumberOfCopies
    ' loop through the pages
    For iPage = 1 To ViewerControl.PageCount
    ViewerControl.PrintPage iPage
    Next iPage
  Next iCopies
  ' end the print job
  ViewerControl.EndDocument
  ' close the printer and delete the temporary font files
  ViewerControl.ClosePrinter
End Sub
```

Entire Document

The following sample shows how to print a document (multiple times). The advantage when with this sample is that the document has to be sent only once, whereas in the previous sample, the pages are sent multiple times, which lowers the performance.

Example:

```
Private Sub PrintDocument_Click()
  ' set amount of copies
  ViewerControl.Copies = 2
  ' print the document
  ViewerControl.PrintDocument "HP LaserJet 4050 Series PS", "MyJob"
End Sub
```

Print to the Default Printer

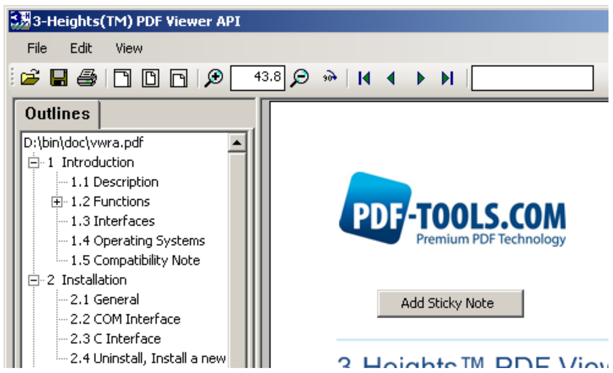
The Windows' default printer can be received using the method GetDefaultPrinter. An alternative way to select the default printer is passing an empty string as the name of the printer.

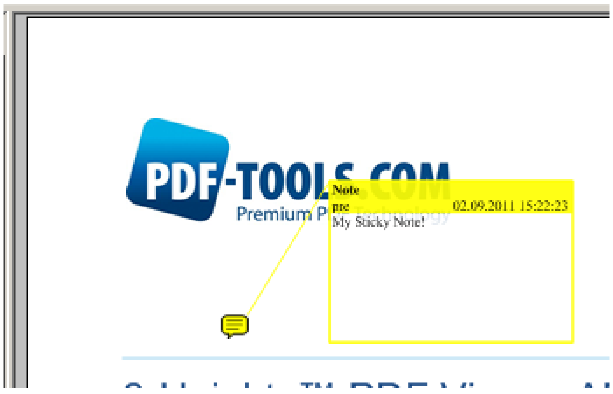
Example:

```
Private Sub PrintToDefaultPrinter_Click()
  ' 1st possibility to print to the default printer:
  PrinterName = ViewerControl.GetDefaultPrinter
  ViewerControl.PrintDocument PrinterName, "MyJob"
  ' 2nd possibility to print to the default printer:
  ViewerControl.PrintDocument "", "MyJob"
End Sub
```

5.4 Sticky Note Annotations

Use the control's property ViewerOptions to enable or disable a right-click context menu "Add Sticky Note" that allows a user to add a sticky note annotation.





The following C# code shows annotations are enabled and disabled.

```
// enable
PdfViewer1.ViewerOptions |=
(int)VIEWEROCXLib.TPDFViewerOption. eViewerOptionEnableAnnotEdit;
// disable
PdfViewer1.ViewerOptions &=
~(int)VIEWEROCXLib.TPDFViewerOption. eViewerOptionEnableAnnotEdit;
```

How to save save annotations is described in the function SaveAs.

6 Pogrammer's Reference

The following section lists all methods and properties of the COM interface of the 3-Heights™ PDF Viewer API.

6.1 Methods

BeginDocument

```
Method Boolean BeginDocument(String DocumentName)
```

Start a new printer job. All pages within one print job are printed successively, e.g. cannot be interrupted by another print job. The printer must previously be chosen with OpenPrinter. During or before the beginning of the print job, a PDF or image document can be opened from file or memory and closed.

The end of the print job is marked with EndDocument.

Parameters:

- DocumentName: The name of the print job

Return value:

- True: Successfully connected to printer and started a print job.
- False: otherwise

Close

```
Method Void Close()
```

Close the currently opened document. If the document is already closed the method does nothing.

ClosePrinter

```
Method Boolean ClosePrinter()
```

Close the connection to the printer. It deletes temporarily installed font files

Return value:

- True: The connection could successfully be closed.
- False: The connection could not be closed

ConvertPt

```
Method Boolean ConvertPt(Long Page, Single x, Single y)
```

Convert document coordinate (for example returned by events) to page coordinates.

0, 0 for document coordinates is the upper left corner of the first page. The border is considered.

Page coordinates are raw (untransformed) user space coordinates of the PDF document. The units are points which is 1/72 inch (A4 = 595x842 points, Letter = 612x792 points). The borders are not part of the page.

Parameters:

- Page: The page number
- x, y: The X and Y coordinates

Return value:

- True: Conversion successful
- False: otherwise

CreateAnnotaion

```
Method Boolean CreateAnnotation(TPDFAnnotation Type)
```

Enable the user to create a new annotation of the specified type using the mouse. This allows an application to offer buttons, menu items or keyboard shortcuts to activate the annotation create mode, e.g. to create a sticky note.

EndDocument

```
Method Boolean EndDocument()
```

Define the end of the printer job. After calling EndDocument, the print job is no longer under the control of the 3-Heights™ PDF Viewer API.

When printing directly to a printer (i.e. not using a spooler), EndDocument means the entire print job is on the printer.

When using the spooler, EndDocument means the entire print job is in the queue of the spooler. It does not imply that is already being printed.

Return value:
- True: The print job was submitted and the connection to the printer could successfully be closed.
- False: otherwise

GetDefaultPrinter

```
Method String GetDefaultPrinter()
```

Return the name of the default printer, if there is a default printer installed on the system. If there is no default printer defined, it returns an empty string.

Return value:
- The name of the default printer

GetPageRect

```
Method Boolean GetPageRect(Long iPageNo, Variant varRect)
```

Get the size of a page in points.

Units: For PDF, 1 point = 1/72 inch; a page of size A4 portrait has normally the values (0, 0, 595, 842), for letter they are (0, 0, 612, 792).

For image documents the resolution of the image is considered. If the image does not have a defined resolution, a default resolution of 96 dpi is assumed, which means 1 point = 1/96 inch.

Parameters:
- iPageNo: The page number, page 1 is the first page in the document
- varRect: The rectangle with the coordinates x1, y1, x2, y2, where the first two represent the position of the lower left corner, and the last two the position of the upper right corner of the page.

Return value:
- True: The page exists and the rectangle is filled
- False: otherwise

See also Example in the function Goto.

GoBack

```
Method Boolean GoBack()
```

When viewing an embedded document, use the GoBack method in order to navigate back to the document in which the embedded document is contained.

Return value:
- True: The parent document was opened successfully
- False: Could not navigate back, e.g. because current document is not an embedded document or user canceled operation due to pending modifications.

Goto

```
Method Boolean Goto(Long iPageNo, Single x, Single y, Single z)
```

Go to a page and position in the document and set the zoom factor.

Parameters:
- iPageNo: The page number, page 1 is the first page in the document
- x: The x-coordinate of the upper left corner
- y: The y-coordinate of the upper left corner
- z: The zoom factor; 1 is 100%, note this is different from the property Zoom, where 100 is 100%

Return value:
- True: The page and position was set successfully
- False: The page or position does not exist

Example: The following VB 6 sample goes to page 2, positions in the middle of the page and sets the zoom factor to half the page width. This means – depending on the shape of the viewer control – the lower right quadrant of page 2 is displayed.

```
' Get size of page 2
Dim r As Variant
If Not ViewerCtrl1.GetPageRect(2, r) Then
  MsgBox "Page 2 does not exist"
End If

' Get the zoom level required to display half the page width
ViewerCtrl1.Page = 2
ViewerCtrl1.FitMode = eFitModeWidth
Dim z As Double
z = ViewerCtrl1.Zoom * 2 / 100 '%

' Set position to middle of 2nd page
If Not ViewerCtrl1.Goto(2, r(0)+(r(2)-r(0))/2, r(1)+(r(3)-r(1))/2, z) Then
  MsgBox "Position does not exist"
End If
```

MarkRectangle

```
Method Void MarkRectangle(Single X, Single Y, Single w, Single h, Long Color, Long
    Width)
```

Define a rectangle at a specified location with a specified color on the foreground of the currently displayed document. For languages that support default parameters, the parameter for the color is optional.

The values of x and y define the location of the bottom left corner in raw (untransformed) PDF user space coordinates. The units are PDF points[3]. The value of w and h define the width and height of the rectangle.

[3] 1 point = 1/72 inch; A4 = 595 by 842 points; letter = 612 by 792 points

Parameters:

- X: The x-coordinate of the lower left corner
- Y: The y-coordinate of the lower left corner
- Width: The width of the rectangle in points
- Height: The height of the rectangle in points
- Color: The color of the marked rectangle as ARGB[4] value. An alpha value of 0 means the marked rectangle is transparent, an alpha value of 255 means it is opaque.
  Compatibility Note: Previous version of the Viewer did not support an alpha channel.
- Width: (optional): This value defines the line width of the rectangle in points. If 0 is specified (default), the rectangle is filled.
  Compatibility Note: In previous version of the Viewer, the line width had different units (1/35 points) or could not be set at all.

Example: Set the color to a green, outlined rectangle.

```
ViewerControl.MarkRectangle 300, 300, 100, 100, &H7F00FF005, 6
```

Open

```
Method Boolean Open(String FileName, String Password)
```

Open a PDF file or raster image file, i.e. make the objects contained in the document accessible. If a document is already open, it is closed first.

The following formats are supported:

- PDF and PDF/A
- Raster image formats such as TIFF, JPEG, JBIG2, JPEG2000, PNG, GIF, PBM, BMP
- FDF

When opening an FDF file, the FDF must contain a path to a PDF file, which can be resolved by the 3-Heights™ PDF Viewer API, which should preferably be an absolute path.

Parameters:

- FileName: The file name and optionally the file path, drive or server string according to the operating systems file name specification rules.
  - Path: e.g. `c:\data\document.pdf`
  - HTTP URL: `http://[username:password@]domain[:port][/resource]`, where `username:password` are used for HTTP basic authentication.
  - HTTPS URL: URL beginning with `https://`
  - FTP URL: URL beginning with `ftp://`
- Password (optional): The user or the owner password of the encrypted PDF document. If this parameter is left out an empty string is used as a default.

Return value:

- True: The file could successfully be opened.
- False: The file does not exist, it is corrupt, or the password is not valid. Use the property ErrorCode for additional information.

OpenMem

```
Method Boolean OpenMem(Variant MemBlock, String Password)
```

Open a PDF document or raster image from memory, i.e. make the objects contained in the document accessible. If a document is already open, it is closed first. For supported formats, see method Open.

[4]Note that Windows uses the order alpha, blue, green, red for ARGB.

Parameters:
- MemBlock: The memory block containing the PDF file given as a one dimensional byte array.
- Password (optional): The user or the owner password of the encrypted PDF document. If this parameter is left out an empty string is used as a default.

Return value:
- True: The document could successfully be opened.
- False: The document could not be opened, it is corrupt, or the password is not valid.

OpenPrinter

```
Method Boolean OpenPrinter(String PrinterName)
```

Open a Printer. Available printers are listed in the "Printers" window of the Windows' "Control Panel".

Parameters:
- PrinterName: The name of the printer. The name is the same as shown on the Settings/Printer window, for example "HP LaserJet 4050 Series PS". It is not the same as the network name. Network printers could look like this: "\\PrinterServer\HP LaserJet 4250 PCL 6".

Return value:
- True: The printer was successful opened.
- False: otherwise

PrintDocument

```
Method Boolean PrintDocument(String PrinterName, String DocumentName, String
    DataType)
```

Print the currently displayed document.

Parameters:
- PrinterName: The name of the printer.
- DocumentName: The name of the print job.
- DataType: The data type of the spool file. It is suggested to use "raw" if printing to a local printer, and to use "emf" printing to a remote printer. See also property DataType.

Return value:
- True: Successfully printed document.
- False: otherwise

PrintDocumentDlg

```
Method Boolean PrintDocumentDlg(String DocumentName)
```

Print the currently displayed document. This will open a printing dialog window that allows the user to select the printer, page ranges and properties.

Parameters:
- DocumentName: The name of the print job.

Return value:
- True: Successfully printed document.
- False: otherwise

PrintPage

```
Method Boolean PrintPage(Long PageNumber)
```

Print a page of the currently opened document. The printer must be selected previously (OpenPrinter). This method must be called after BeginDocument and before EndDocument.

Parameters:
- PageNumber: This is the number of the page in PDF file to be printed.

Return value:
- True: The page was successfully printed.
- False: otherwise

SaveAs

```
Method Boolean SaveAs(String FileName, String UserPw, String OwnerPw,
    TPDFPermission PermissionFlags, Long KeyLength)
```

Save the currently opened document.

The target file format must be the same as the file format of the currently opened document. An exception is FDF: If a PDF document is opened, annotations (sticky notes) can be added. The result can be saved as PDF or FDF. If saved as FDF, it contains all annotations, not only those added by the 3-Heights™ PDF Viewer API, but also those contains in the original PDF document.

Parameters:
- FileName: The file name and optionally the file path, drive or server string according to the operating systems file name specification rules.
  When saving an FDF, this can be either a file path or an URL. When an URL is provided the FDF is uploaded to the server using HTTP put.
  The file type is defined via file extension.
- UserPwd (optional): Set the user password of the PDF document. If this parameter is omitted, the default password is used. Use 0 to set no password.
- OwnerPwd (optional): Set the owner password of the PDF document. If this parameter is omitted, the default password is used. Use 0 to set no password.
- PermissionFlags (optional): The permission flags. By default no encryption is used (-1). The permissions that can be granted are listed at the enumeration TPDFPermission.
  To not encrypt the output document, set PermissionFlags to ePermNoEncryption (-1), user and owner password to 0.
  In order to allow high quality printing, flags ePermPrint and ePermDigitalPrint need to be set.
- KeyLength (optional): Default set to 128. With this option the length of the encryption key can be set. Supported are all values from 40 to 128 that are multiples of 8. The two most commonly used values are 40 and 128.[5]
  The default value is calculated based on the selected permission flags.

Return value:
- True: The opened document could successfully be saved to file.
- False: Otherwise. One of the following occurred[6]:
  - The output file cannot be created

SearchFirst

```
Method Boolean SearchFirst(String SearchText)
```

Mark the SearchText in the viewer if the string can be found.

[5] 128 bit requires PDF 1.4 or later (Acrobat 5). Acrobat only supports 40 and 128 bit encryption. Other tools, such as the 3-Heights™ tools also support other encryption key lengths.

[6] This is not a complete list. If SaveAs returs `False`, it is recommended to abort the processing of the file and log the error code and error message.

Parameters:
- SearchText: The string to be searched for. It may not contain blanks.

Return value:
- True: Search was string found.
- False: No occurrence of the search string was found.

SearchNext

```
Method Boolean SearchNext()
```

The method SearchFirst marks the first instance of the string, every subsequent call to SearchNext jumps to the next instances.

Return value:
- True: A further occurrence of the search string was found.
- False: No further occurrence of the search string was found.

SetAnnotPropertiesStr

```
Method Boolean SetAnnotPropertiesStr(String Properties)
```

Set non-overridable default values for the creation of new annotations. For example, this is to ensure that a user can create annotations under his own name only.

SetCMSEngine

```
Method Boolean SetCMSEngine(String CMSEngine)
```

Set the Color Management System (CMS) Engine. The following strings are supported:

- "None": No CMS is applied. This results in the maximum possible contrast.
- "Neugebauer": The Neugebauer algorithm efficiently converts CMYK to RGB. It does not need any color profiles. The results, however, look similar to conversion using color profiles.
- "MSICM": The Microsoft ICM Engine is the default engine.
- "lcms": lcms is mostly used on Unix platforms.

SetCursor

```
Method Boolean SetCursor(TPDFCursorMode CursorMode, LONG Cursor)
```

Set the cursor of a specific cursor mode.

Parameters:
- CursorMode: The cursor mode, for which the new cursor should be set.
- Cursor: The new cursor may be any of the standard resource identifiers supported by the Windows LoadCursor function.

Example

```
ViewerControl.SetCursor(/* eCursorModeMove        */ 0, /* IDC_HAND  */ 32512)
ViewerControl.SetCursor(/* eCursorModeLink        */ 4, /* IDC_ARROW */ 32649)
```

SetUILanguage

```
Method Boolean SetUILanguage (String Language)
```

Set the language of the user interface of the viewer control.

Supported languages are:

`en`	English
`de`	German
`ja`	Japanese

By default, the locale of your application is used. If this is not available, the fallback is English.

6.2 Properties

Border

```
Property Long Border
Accessors: Get, Set
Default: 300
```

The PDF Viewer control displays pages on top of a gray background. The border defines the width between the edge of the control and the page, as well as the distance between the pages in the continuous page mode. This property must be set before opening a document.

Center

```
Property Boolean Center
Accessors: Get, Set
Default: False
```

Set or get the center mode. When set to true, the document is horizontally and vertically centered on the page. When set to false, the document is printed to the upper left corner of the page.

Copies

```
Property Integer Copies
Accessors: Get, Set
Default: -1
```

Set or get the number of copies. This property should be used in combination with PrintDocument. If the value of this property is set to -1, the number of copies defined in the printer is applied.

CursorMode

```
Property TPDFCursorMode CurserMode
Accessors: Get, Set
Default: eCursorModeMove
```

Set or get the current cursor mode. There are five supported cursor modes. The default mode is the move mode. Most events can be caught in the mode eCurserModeNoop. The eCursorModeMark can be used to catch the event method OnMarkRect. See also enumeration TPDFCursorMode.

DataType

```
Property String DataType
Accessors: Get, Set
Default: ""
```

Set or get the data type of the spool file. There are two valid data types: “raw” and “emf”.

`"raw"`	“raw” is with respect to the printer language. E.g. if “raw” is used for a PCL printer, a PCL file is created, if “raw” is used for a PS printer, a PS file is created. For local printers the DataType should be set to “raw”.
`"emf"`	If “emf” is used, an EMF[7] file is generated. The EMF file can be sent over a network and at its destination the (remote-) printer driver converts it to a raw file. For network printers “emf” should be used and it comes with the following two advantages: ▪ It takes less bandwidth to send the spool file over the network, because and EMF file is smaller than raw spool file. ▪ The workload is balanced: On the host where the Printer API resides, the EMF file is generated, on the host where the printer resides, the EMF file is converted to a raw file.
`""`	If DataType is set to an empty string, the data type is inherited from the printer’s setting of the current user. In any situation where the current user settings are not well defined (e.g. IIS), the DataType should be set explicitly to either “raw” or “emf”.

DefaultSource

```
Property Integer DefaultSource
Accessors: Get, Set
Default: -1
```

The default source defines from which input tray the paper shall be selected. For default values see Appendix Paper Bins. There is no property to set the output paper tray. In order to set the output paper try, use the device mode functionality.

DPI

```
Property Long DPI
Accessors: Get, Set
Default: 108*
```

Get or set the resolution in dots per inch (DPI) in X and Y. See also XDPI and YDPI.

Duplex

```
Property Integer Duplex
Accessors: Get, Set
Default: -1
```

Set or get the duplex mode. For Windows default values, see Appendix Duplex Modes. It is suggested to use the default values 1, 2 and 3.

-1	Use printer default
1	Simplex
2	Vertical Duplex
3	Horizontal Duplex

[7] Enhanced Metafile is a graphic format from Microsoft.

ErrorCode

```
Property TPDFErrorCode ErrorCode
Accessors: Get
```

This property can be accessed to receive the latest error code. See also enumeration TPDFErrorCode. PDF-Tools error codes are listed in the header file *pdferror.h*. Please note, that only few of them are relevant for the 3-Heights™ PDF Viewer API.

FitMode

```
Property TPDFFitMode FitMode
Accessors: Get, Set
Default: eFitModeWidth
```

The fit mode defines how the pages are displayed in the viewer component. There are three supported fit modes:

0	Display actual size
1	Fit the page
2	Fit the width

The fit mode can be altered at any time.

FitPage

```
Property Boolean FitPage
Accessors: Get, Set
Default: False
```

The fit-page property defines how the PDF page should fit the paper size. Allowed values are:

`True`	The page is resized so that both, page width and height fit on the printable part of the paper supported by the printer device. The ratio width to height remains unchanged.
`False`	The size of the page remains unchanged. If part of the content is outside the printable area (i.e. close to the border of the page) it will not be printed.

LayoutMode

```
Property TPDFLayoutMode LayoutMode
Accessors: Get, Set
Default: eLayoutDocument
```

The layout mode determines whether pages are displayed continuously or not. Consequently there are two available layout modes, these are: eLayoutModeDocument and eLayoutModePage.

The layout mode may be changed whilst a document is being displayed. The layout mode has no impact on printing.

Modified

```
Property Boolean Modified
Accessors: Get
Default: False
```

This property indicates, whether the document contains unsaved changes, e.g. because its annotations have been edited.

OffsetX,OffsetY

```
Property Long OffsetX
Property Long OffsetY
Accessors: Get, Set
Default: 0
```

Set or get the X and Y-offset of the page on the paper. Units: 1/100 millimeters.

Options

```
Property TPDFRendererOption Options
Accessors: Get, Set
Default: eOptionBanding + eOptionBicubic + eOptionHighQuality
```

Set or get a specific option.

Use bitwise "OR" to add an option.

Use bitwise "AND NOT" to remove an option.

See also enumeration TPDFRendererOption.

Orientation

```
Property Integer Orientation
Accessors: Get, Set
Default: -1
```

Set or get the orientation of the paper. Allowed values are:

`-2`	Automatic
`-1`	Use printer setting (default)
`1`	Force portrait
`2`	Force landscape

When not specified, the PDF Printer API uses the setting of the printer.

Compatibility Note: In older versions, the default was set to automatic, which places the page so on the paper that it fits best.

Page

```
Property Long Page
Accessors: Get, Set
Default: 0
```

Set or get the page in the currently opened document. The page is automatically updated by the control.

PageAtPos

```
Property Long PageAtPos (fRelPos As Single) Accessors: Get
```

Get the page at a relative view port position (0: top, 1: bottom). This property can be used to detect which pages are displayed, if multiple pages are displayed and the property Page may not be accurate enough.

Example:

`PageAtPos(0) = 4`

`PageAtPos(0.25) = 5`

```
PageAtPos(0.5) = 5
PageAtPos(0.75) = 5
PageAtPos(1) = 6
```

The above values indicate that the pages 4, 5 and 6 are displayed. It is likely that only page number 5 is fully displayed. From page 4 the bottom, and from page 6 the top are displayed.

More precisely: At the very top of the control, page 4 is displayed (this is also returned by the property Page). However short after that (in the uppermost 25% of the control) page 5 starts. Page 5 is displayed at 25%, in the middle of the control and at 75%. At the bottom the top of page 6 is shown.

PageCount

```
Property Long PageCount
Accessors: Get
```

Return the number of pages of an open document. If the document is closed then 0 is returned.

PagesPerWindow

```
Property Integer PagesPerWindow
Accessors: Get, Set
Default: 1
```

The viewer control can display 1 or 2 pages per window next to each other. The default is 1.

PanX

```
Property Long PanX
Accessors: Get, Set
Default: 0
```

This value represents the position of the horizontal scrollbar. Its units are 1/100 millimeters.

It is 0 at the very left position. If the viewer control is wider than the width of the currently displayed page, the PanX becomes negative representing the distance between the left border of the current page and the left border of the viewer control.

PanY

```
Property Long PanY
Accessors: Get, Set
Default: 0
```

This value represents the position of the vertical scrollbar. Its units are 1/100 millimeters.

Using LayoutMode = eLayoutPage, it is 0 at the top of the current page. Using LayoutMode = eLayoutDocument, it is 0 at the top of the first page of the document. If the viewer control is higher than the height of the currently displayed page, the PanY becomes negative representing the distance between the upper border of the page and the upper border of the viewer control.

PaperSize

```
Property Integer PaperSize
Accessors: Get, Set
Default: -1
```

Get or set the paper size. The 118 Windows default paper sizes are listed in the Appendix Paper Sizes.

RenderingMode

```
Property TPDFRenderingMode RenderingMode
Accessors: Get, Set
Default: eModeFast
```

Set or get the rendering mode. The supported rendering modes are listed in the enumeration TPDFRendering-Mode. The default and recommended mode is `eModeFast`.

Rotate

```
Property Integer Rotate
Accessors: Get, Set
Default: 0
```

Rotate pages, a positive number turns the page clockwise. The value must be a multiple of 90, i.e. valid values are -270, -180, -90, 0, 90, 180 and 270. This property is reset 0 when opening a document.

SizeX,SizeY

```
Property Long SizeX
Property Long SizeY
Accessors: Get, Set
Default: 0
```

Even though paper sizes can be set directly in millimeters using these properties, is suggested to use the property PaperSize instead.

TreeViewWidth

```
Property Long TreeViewWidth
Accessors: Get, Set
Default: 200
```

The viewer control provides an optional navigation control on the left hand side. This navigation control can contain a tree view of the outlines[8] or a page list. The width of this navigation control can be set in pixel using this property.

ViewerOptions

```
Property TPDFViewerOptions ViewerOptions
Accessors: Get, Set
Default: eViewerOptionScrollbars + eViewerOptionOutlines +
    eViewerOptionDoubleBuffer
```

Get or set the viewing options. See enumeration TPDFViewerOption.

XDPI

```
Property Single XDPI
Accessors: Get, Set
Default: *calculated*
```

Get or set the resolution in the X-axis in DPI[9].

The default value is calculated and depends on the size of the monitor and resolution. In order for the value to be calculated correctly, it is required that the correct monitor driver is installed. On a 4:3 17 inch monitor with a resolution of 1280x1024, this value is approximately 102[10].

A potential reason why this value could be calculated falsely is an incorrectly installed monitor driver.

[8]Outlines are sometimes referred to as "Bookmarks"
[9]DPI = dots per inch
[10]Note that in Adobe Acrobat this is a fixed value of 96 for X and Y direction.

YDPI

```
Property Single YDPI
Accessors: Get, Set
Default: *calculated*
```

Get or set the resolution in the Y-axis in DPI.

The default value is calculated and depends on the size of the monitor and resolution. In order for the value to be calculated correctly, it is required that the correct monitor driver is installed. On a 4:3 17 inch monitor with a resolution of 1280x1024, this value is approximately 108.

See also XDPI.

Zoom

```
Property Single Zoom
Accessors: Get, Set
Default: 100
```

The zoom level defines how large the document is displayed with respect to its true size. In order to see a document at its true size at 100% zoom, it is required that the properties for XDPI and YDPI are correctly set.

6.3 Events

There is a series of events which can be caught using the corresponding events handlers. Mouse related events depend on the curser mode. They will not fire in all curser modes. In the eCurserModeNoop most will fire. OnMarkRect fires in eCurserModeMark.

```
Event OnChar(Long iChar, Long iFlags)
```

```
Event OnGoto(String bstrType, Long PageNo, Single x, Single y, Single z)
Event OnGotoR(String bstrFileName)
Event OnGotoE(String bstrName, String bstrTmpName, String bstrType, Long PageNo,
    Single x, Single y, Single z)
```

```
Event OnKeyDown(Long iVirtKey, Long iFlags)
Event OnKeyUp(Long iVirtKey, Long iFlags)
```

```
Event OnLButtonDblClk(Single x, Single y)
Event OnLButtonDown(Single x, Single y)
Event OnLButtonUp(Single x, Single y)
```

```
Event OnMarkRect(Single Left, Single Top, Single right, Single bottom)
```

```
Event OnMouseMove(Single x, Single y)
```

```
Event OnPage(Long iPage)
```

```
Event OnRButtonDblClk(Single x, Single y)
Event OnRButtonDown(Single x, Single y)
Event OnRButtonUp(Single x, Single y)
```

```
Event OnURI(Single bstrURI)
Event OnZoom(Single fZoom)
Event OnZoomRect(Single Left, Single Top, Single right, Single bottom)
```

```
Event OnCursorModeChanged(TPDFCursorMode oldCursorMode, TPDFCursorMode
    newCursorMode)
```

6.4 Enumerations

Note: Depending on the interface, enumerations may have “TPDF” as prefix (COM, C) or “PDF” as prefix (.NET) or no prefix at all (Java).

TPDFCursorMode

`eCursorModeMove`	Allow using the mouse to scroll (default)
`eCursorModeZoom`	Mark rectangle and zoom into it
`eCursorModeMark`	Mark a rectangle, see event method OnMarkRect.
`eCursorModeNoop`	No operation mode
`eCursorModeLink`	Allow clicking on links

TPDFErrorCode

All TPDFErrorCode enumerations start with “PDF_” followed by a single letter which is one of “S”, “E”, “W” or “I”, an underscore and a descriptive text. The single letter gives in an indication of the type of error. These are: Success, Error, Warning, Information. With respect to corrupt PDF files: An error indicates a corruption in the PDF, the file may or may not be readable. A warning indicates the file is readable but not valid.

A full list of all PDF Tools error codes is available in the header file *pdferror.h*. Note that only a few are relevant for the PDF Viewer API. The most common are listed here.

`PDF_S_SUCCESS`	The operation was completed successfully.
`LIC_E_NOTSET,` `LIC_E_NOTFOUND, …`	Various license management related errors.
`PDF_E_FILEOPEN`	Failed to open the file.
`PDF_E_FILECREATE`	Failed to create the file.
`PDF_E_PASSWORD`	The authentication failed due to a wrong password.
`PDF_E_UNKSECHANDLER`	The file uses a proprietary security handler, e.g. for a proprietary digital rights management (DRM) system.
`PDF_E_XFANEEDSRENDERING`	The file contains unrendered XFA form fields, i.e. the file is an XFA and not a PDF file.

TPDFFitMode

`eFitModeActualSize`	Display actual size
`eFitModePage`	Display whole page
`eFitModeWidth`	Display width of current page (default)

TPDFLayoutMode

`eLayoutPage`	Display one page at a time
`eLayoutDocument`	Display all pages

TPDFPermission

An enumeration for permission flags. If a flag is set, the permission is granted.

`ePermNoEncryption`	Do not apply encryption. This enumeration shall not be combined with another enumeration. When using this enumeration set both passwords to an empty string or null.
`ePermAll`	Grant all Permissions
`ePermPrint`	Low resolution printing
`ePermModify`	Changing the document
`ePermCopy`	Content copying or extraction
`ePermAnnotate`	Annotations
`ePermFillForms`	Filling of form fields
`ePermSupportDisabilities`	Support for disabilities
`ePermAssemble`	Document Assembly
`ePermDigitalPrint`	High resolution printing

Changing permissions or granting multiple permissions is done using a bitwise or operator. Changing the current permissions in Visual Basic should be done like this:

Allow Printing:	`Permission = Permission Or ePermPrint`
Prohibit Printing:	`Permission = Permission And Not ePermPrint`

TPDFRendererOption

Renderer options are set using the property Options. To combine multiple options use a bitwise OR operator. To disable an option use the bitwise AND NOT operators.

Example: Visual Basic

Enable or disable an option, and leaving all other options untouched:

```
'Enable High Quality
.Options = .Options OR eOptionHighQuality
'Disable High Quality
.Options = .Options AND NOT eOptionHighQuality
```

Example: C/C++

```
int iOptions = PdfViewerGetViewerOptions(hViewer);
// Enable High Quality
PdfViewerSetViewerOptions(hViewer, iOptions | eOptionHighQuality);
// Disable High Quality
PdfViewerSetViewerOptions(hViewer, iOptions & ~eOptionHighQuality);
```

The following list includes enumerations that are relevant for the 3-Heights™ PDF Viewer API. Note that there are more enumerations available, but they are unrelated to this API.

`eOptionBanding`	The data is sent in small chunks. This is mainly for older printers with limited memory.
`eOptionBilinear`	A bilinear image filter is applied to images to improve the image quality. This option cannot be combined with eOptionBicubic.
`eOptionBicubic`	(default) A bi-cubic image filter is applied to images to improve the image quality. This option cannot be combined with eOptionBilinear. The highest quality is obtained by combining eOptionHighQuality and eOptionBicubic, this combination also requires to most CPU power.
`eOptionBitmap`	The pages are rendered in a bitmap, which then is sent to the device.
`eOptionDoNotPrintSig`	Do not print digital signatures.
`eOptionDisablePatterns`	Disable patterns.
`eOptionDisablePS`	Disable direct PostScript injection.
`eOptionJPEG`	If a printer supports JPEG compression, then the pages can be sent with JPEG compression to reduce the size of the spool file.
`eOptionHighQuality`	(default) Anti-aliasing for text and path objects and filtering of image objects can be turned off and on with this option.
`eOptionOpenType`	Convert embedded fonts to Open Type fonts.
`eOptionNoEmbedded`	Do not use embedded fonts. Instead fonts from the operating system's font directory are used (%Systemroot%\fonts).
`eOptionOutlines`	Convert fonts into vector graphics.
`eOptionPNG`	If a printer supports PNG compression, then the pages can be sent with PNG compression to reduce the size of the spool file.
`eOptionPreInstalled`	Replace embedded fonts with a pre-installed font if the same font is already installed on the OS.
`eOptionPrintOnlySig`	Print the digital signature appearance only (without any status appearances, e.g. valid or invalid).
`eOptionTransparency`	(default) Transparency is applied in memory. The blended result is then sent to the device.
`eOptionTrueType`	CFF and Type1 fonts are converted to True Type fonts. This option overrules option *eOptionType1*.
`eOptionType1`	CFF fonts are converted to Type1 fonts.
`eOptionUseFastImages`	Always print images in fast mode. This should help resolving performance issues with complex images and image masks of documents that are to be printed in accurate mode.
`eOptionWindows9x`	Run in Windows 9x compatibility mode. Setting this option can resolve issues with nested transformation matrices which are not supported by all devices.

TPDFRenderingMode

`eModeAccurate`	The accurate mode is intended for virtual printers such as a TIFF printer. It uses the Windows GDI+ for rendering. This mode allows for image filtering, sub-pixel rendering and anti-aliasing. It should not be applied for physical devices, such as a laser printer, due to the fact that those devices do not support the above features. Using the accurate mode creates generally larger spool files than the fast mode.
`eModeDirect`	This mode is deprecated.
`eModeFast`	The fast mode is the recommended mode for printing to any *physical printer device* such as a laser printer, or ink jet printer. It uses the Windows GDI for rendering. This mode is generally faster and creates smaller spool files than the accurate mode. Use this mode for high resolution (600 dpi).

TPDFRotateMode

`eRotateName`	Do not rotate the page; do not consider the viewing rotation attribute of the PDF page.
`eRotateAttribute`	Set the rotation to the viewing rotation attribute of the PDF page, i.e. rendering the page with the same rotation as it is displayed in a PDF Viewer.
`eRotatePortrait`	Rotate page to portrait.
`eRotateLandscape`	Rotate page to landscape.

TPDFViewerOption

`eViewerOptionDisableGoto`	Disable GoTo links (links within this document)
`eViewerOptionDisableGotoR`	Disable GoToR links (links outside this document)
`eViewerOptionDisableURI`	Disable URI links (web links)
`eViewerOptionDisableLinks`	Disable all types of interactive links within PDF document (GoTo, GoToR, URIs).
`eViewerOptionDisableMouseWheel`	Disable mouse wheel for scrolling
`eOptionEnableAnnotEdit`	If this option is set the user interface offers commands to add, delete and edit text annotations (sticky notes).
`eViewerOptionAnnotations`	Enables pane for displaying and modifying annotations.
`eViewerOptionHighlighting`	Show the highlighting menu on the left hand side to allow for searching and marking different texts.
`eViewerOptionScrollbars`	Show scrollbars (default)

`eViewerOptionOutlines`	Show outlines (also known as Bookmarks) (default)
`eViewerOptionPages`	Show thumbnails. Each page is represented as a pre-view image. Enabling this option can reduce the performance. (Compatibility Note: Versions prior to 1.9.10.1 did not support displaying thumbnails, instead they displayed a page number tree.)
`eViewerOptionDoubleBuffer`	Enable double buffering Double buffering is a technique used to minimize visible artifacts from the drawing process (flickering). As opposed to single buffering, the memory used for the displayed page is kept separately for writing and reading. This results in a better viewing quality but also requires more memory and CPU.
`eViewerOptionEmbeddedFiles`	Show the list of embedded files on the left hand side.
`eViewerOptionSaveEF`	Allows the user to save embedded files.

7 Tips, Tricks, and Troubleshooting

7.1 Performance

The 3-Heights™ PDF Viewer API provides a variety of settings to tune the performance. In most cases a simple rules applies: Higher quality takes more resources (memory, CPU) and therefore lowers the performance.

The following settings have an impact on the performance:

- Content of the PDF: A document with thousands of objects requires more time for rendering than a page with plain text.
- Resolution: The higher the size of the viewer window the more pixels need to be drawn and the lower the performance.
- Double Buffering: Enabling double buffering requires extra memory and CPU.
- Rendering Mode: Disabling the accurate rendering mode improves the performance at the cost of visual quality. It should only be applied at high zoom levels or high resolutions.
- Filters: The filters for “Bilinear”, “Bicubic” and “HighQuality” require extra CPU. Thumbnails: Displaying thumbnails reduces the performance. Use outlines instead.
- Thumbnails: Displaying thumbnails reduces the performance. Use outlines instead.

7.2 Viewing vs Printing

First and foremost, once needs to distinguish between incorrect fonts when viewing the document in the viewer control and incorrect fonts when printing to paper using the print functionality of the viewer control. This chapter is about incorrect fonts when viewing the document. If you have correct fonts when viewing, but incorrect fonts when printing, the best choice is probably to update or alter the printer driver.

7.3 Handle Non-Embedded Fonts

Font Replacement Strategy

This section describes the exact behavior of font handling of the rendering engine. It is rather technical and it is not required to be understood in order to properly use the software.

The following steps are performed sequentially in the search of a font. If a font is found, the search is stopped; otherwise the next step is performed.

1. If the font is not embedded or eOptionPreInstalled is set:
   a. If the font name appears in the [Replace] section in the configuration file "fonts.ini" the name is replaced and looked up in the installed font collection
   b. if it is a standard font[11] it is replaced by the equivalent TrueType font name and it is looked up in the installed font collection
   c. If the font name appears in the [Fonts] section in the configuration file "fonts.ini" the name is replaced and looked up in the installed font collection
   d. If the font has "Italic" or "Bold" in its name the font without these styles is looked up in the installed font collection
2. If a font name is looked up in the installed font collection then the name compare is performed as follows:
   a. PostScript name
   b. TrueType name without blanks (a missing style is interpreted as "Regular" or "Normal")
   c. TrueType name without modifications
3. If the font is embedded, it is converted to a Windows compatible font and temporarily installed. If eOptionNoEmbedded is used then the glyphs of the fonts are converted to either bitmaps or outlines[12]. If eOptionUseOutlines is used then the glyphs are converted to outlines only.
4. If the font is not embedded and the Unicodes are available then the nearest font from the installed font collection is tailored to the metrics of the font.
5. If the font is embedded then it is converted to outlines.
6. In all other cases the nearest font from the installed font collection is used

Installed Font Collection

The installed font collection contains fonts from the directories *%SystemRoot%\Fonts* and "Fonts", which must be a direct sub-directory of where the product's executable resides. The fonts and their properties are cached in a font cache, located in the files *%TMP%\font-database**. It is recommended to clear the cache, if you add or remove fonts from the font directories.

Using the Font Mapping File *fonts.ini*

"fonts.ini" is a configuration file to map fonts used in the PDF to fonts pre-installed on the system. The mapping file must reside a directory named "Fonts", which must be a direct sub-directory of where the main DLL or executable resides. The mapping file is optional. It consists of two sections: [fonts] and [replace]. Both sections are used to map fonts in the PDF to fonts in the installed font collection on the operating system. This comes into play when the font in the PDF document does not have an embedded font program, or the embedded font is not usable. The mapping only works if the font types of the specified fonts are matching; e. g. if the font in the PDF is a symbolic font, such as "Symbol" or "ZapfdingBats", the mapped font must be symbolic too. The section [fonts] is only considered if the font-matcher does not find an appropriate font amongst the existing installed fonts. It is suggested to only use this section. The section [replace] is stronger and applied before the font-matcher. This means a font will be replaced as defined, even if the correct installed font is available on the system.

Syntax:
The syntax of the mapping file is this:

```
[fonts]
PDF_ font_ 1=installed_ font_ 1, {font_ style}
```

[11] e.g. Times-Roman, Helvetica, Courier
[12] The outline of a glyph is a vector graphic without any reference to the original font program.

```
PDF_ font_ 2=installed_ font_ 2, {font_ style}
[ replace ]
PDF_ font_ n=installed_ font_ n, {font_ style}
```

PDF_font_* is the name of the font in the PDF. This name can be found in one of the following ways:

- Use any tool that can list fonts. Such as 3-Heights PDF Extract or 3-Heights PDF Optimization. Ignore possible prefixes of subset fonts. A subset prefix consists of 6 characters followed by the plus sign. For example "KHFOKE+MonotypeCorsiva", in this case only use "MonotypeCorsiva" as font name in the mapping file.
- Open the document with Adobe Acrobat, use the "MarkUp Text Tool" , mark the text of which you would like to know the font name, right-click it, select "Properties..."

installed_font_* is the font family name of the installed font. To retrieve this name, find the font in the Windows' font directory and open it by double-clicking. The first line in the property window displays the font family name (this may vary depending on the operating system). The font family name does not include font styles; so an example of a font family name is "Arial", but not "Arial Italic".

font_style is an optional style, that is added coma-separated after the font family name. The style is always one word. Examples of font styles are "Italic", "Bold", "BoldItalic".

Example:

```
[fonts]
Ryumin-Light=MS Mincho
GothicBBB-Medium=MS Gothic
[replace]
ArialIta=Arial,BoldItalic
```

Other Ways to Deal with Text Issues

The following list provides possible work-arounds if text is printed incorrectly. Options should be tried in ascending order.

1. Using the option (`eOptionNoEmbedded`) inhibits all embedded fonts from being used in the spool file and the printer hardware. Instead the glyphs are converted to either bitmaps or outlines. Using the option (`eOptionOutline`) at the same time the conversion is restricted to outlines.
2. Using the option (`eOptionPreInstalled`) inhibits embedded fonts which have the same name as the corresponding installed font from being used. This option can also be used to reduce the number of fonts in a spool file if the printer hardware memory capacity is limited.
3. Pre-render the page in a bitmap and send the pre-rendered image to the printer (`eOptionBitmap`). This results in large spool files.

7.4 Transparency

The 3-Heights™ rendering engine supports transparency functions such as a number of blend modes as well as isolated and non-isolated transparency groups, but not transparency in general. Certain types of tiling and shading patterns may not correctly be reproduced by the rendering engine.

8 Licensing and Copyright

The 3-Heights™ PDF Viewer API is copyrighted. This user's manual is also copyright protected; it may be copied and given away provided that it remains unchanged including the copyright notice.

9 Contact

PDF Tools AG

Kasernenstrasse 1

8184 Bachenbülach

Switzerland

http://www.pdf-tools.com

A Default Values

A.1 Duplex Modes

1	Simplex
2	Vertical Duplex
3	Horizontal Duplex

A.2 Paper Bins

1	Upper
2	Lower
3	Middle
4	Manual
5	Envelope
6	Envelope Manual
7	Auto
8	Tractor
9	Small FMT
10	Large FMT
11	Large Capacity
12	undef.
13	undef.
14	Cassette
15	From Source

A.3 Paper Sizes

1	Letter 8 1/2 x 11 in
2	Letter Small 8 1/2 x 11 in
3	Tabloid 11 x 17 in
4	Ledger 17 x 11 in
5	Legal 8 1/2 x 14 in
6	Statement 5 1/2 x 8 1/2 in
7	Executive 7 1/4 x 10 1/2 in
8	A3 297 x 420 mm
9	A3 297 x 420 mm
10	A4 Small 210 x 297 mm
11	A5 148 x 210 mm
12	B4 (JIS) 250 x 354
13	B5(JIS)182x257mm
14	Folio 8 1/2 x 13 in
15	Quarto 215 x 275 mm
16	10x14 in
17	11x17 in
18	Note 8 1/2 x 11 in
19	Envelope # 9 3 7/8 x 8 7/8
20	Envelope # 10 4 1/8 x 9 1/2
21	Envelope # 11 4 1/2 x 10 3/8
22	Envelope # 12 4 \276 x 11
23	Envelope # 14 5 x 11 1/2
24	C size sheet
25	D size sheet
26	E size sheet
27	Envelope DL 110 x 220mm
28	Envelope C5 162 x 229 mm
29	Envelope C3 324 x 458 mm
30	Envelope C4 229 x 324 mm
31	Envelope C6 114 x 162 mm
32	Envelope C65 114 x 229 mm
33	Envelope B4 250 x 353 mm
34	Envelope B5 176 x 250 mm
35	Envelope B6 176 x 125 mm
36	Envelope 110 x 230 mm
37	Envelope Monarch 3.875 x 7.5 in
38	63/4Envelope35/8x61/2in
39	USStdFanfold147/8x11in
40	German Std Fanfold 8 1/2 x 12 in
41	German Legal Fanfold 8 1/2 x 13 in
42	B4 (ISO) 250 x 353 mm
43	Japanese Postcard 100 x 148 mm
44	9 x 11 in
45	10 x 11 in
46	15 x 11 in
47	Envelope Invite 220 x 220 mm
48	RESERVED–DO NOT USE
49	RESERVED–DO NOT USE
50	Letter Extra 9 \275 x 12 in
51	Legal Extra 9 \275 x 15 in
52	Tabloid Extra 11.69 x 18 in
53	A4 Extra 9.27 x 12.69 in

54	Letter Transverse 8 \275 x 11
55	A4 Transverse 210 x 297 mm
56	Letter Extra Transverse 9\275
57	SuperA/SuperA/A4 227 x 356
58	SuperB/SuperB/A3 305 x 487
59	Letter Plus 8.5 x 12.69 in
60	A4 Plus 210 x 330 mm
61	A5 Transverse 148 x 210 mm
62	B5 (JIS) Transverse 182 x 257 mm
63	A3 Extra 322 x 445 mm
64	A5 Extra 174 x 235 mm
65	B5 (ISO) Extra 201 x 276 mm
66	A2 420 x 594 mm
67	A3 Transverse 297 x 420 mm
68	A3 Extra Transverse 322 x 445 mm
69	Japanese Double Postcard 200 x 148 mm
70	A6 105 x 148 mm
71	Japanese Envelope Kaku # 2
72	Japanese Envelope Kaku # 3
73	Japanese Envelope Chou # 3
74	Japanese Envelope Chou # 4
75	Letter Rotated 11 x 8 1/2 11 in
76	A3 Rotated 420 x 297 mm
77	A4 Rotated 297 x 210 mm
78	A5 Rotated 210 x 148 mm
79	B4 (JIS) Rotated 364 x 257 mm
80	B5 (JIS) Rotated 257 x 182 mm
81	Japanese Postcard Rotated 148 x 100 mm
82	Double Japanese Postcard Rotated 148 x 200 mm
83	A6 Rotated 148 x 105 mm
84	Japanese Envelope Kaku # 2 Rotated
85	Japanese Envelope Kaku # 3 Rotated
86	Japanese Envelope Chou # 3 Rotated
87	Japanese Envelope Chou # 4 Rotated
	88B6(JIS)128x182mm
89	B6 (JIS) Rotated 182 x 128 mm
90	12x11in
91	Japanese Envelope You # 4
92	Japanese Envelope You # 4 Rotated
93	PRC 16K 146 x 215 mm
94	PRC 32K 97 x 151 mm
95	PRC 32K(Big) 97 x 151 mm
96	PRC Envelope # 1 102 x 165 mm
97	PRC Envelope # 2 102 x 176 mm
98	PRC Envelope # 3 125 x 176 mm
99	PRC Envelope # 4 110 x 208 mm
100	PRC Envelope # 5 110 x 220 mm
101	PRC Envelope # 6 120 x 230 mm
102	PRC Envelope # 7 160 x 230 mm
103	PRC Envelope # 8 120 x 309 mm
104	PRC Envelope # 9 229 x 324 mm
105	PRC Envelope # 10 324 x 458 mm

106	PRC 16K Rotated
107	PRC 32K Rotated
108	PRC 32K(Big) Rotated
109	PRC Envelope # 1 Rotated 165 x 102 mm
110	PRC Envelope # 2 Rotated 176 x 102 mm
111	PRC Envelope # 3 Rotated 176 x 125 mm
112	PRC Envelope # 4 Rotated 208 x 110 mm
113	PRC Envelope # 5 Rotated 220 x 110 mm
114	PRC Envelope # 6 Rotated 230 x 120 mm
115	PRC Envelope # 7 Rotated 230 x 160 mm
116	PRC Envelope # 8 Rotated 309 x 120 mm
117	PRC Envelope # 9 Rotated 324 x 229 mm
118	PRC Envelope # 10 Rotated 458 x 324 mm