3-Heights™ PDF Extract API

Version 4.5

User Manual

Contact: pdfsupport@pdf-tools.com

Owner: **PDF Tools AG**

Kasernenstrasse 1
8184 Bachenbülach
Switzerland

Switzerland

http://www.pdf-tools.com

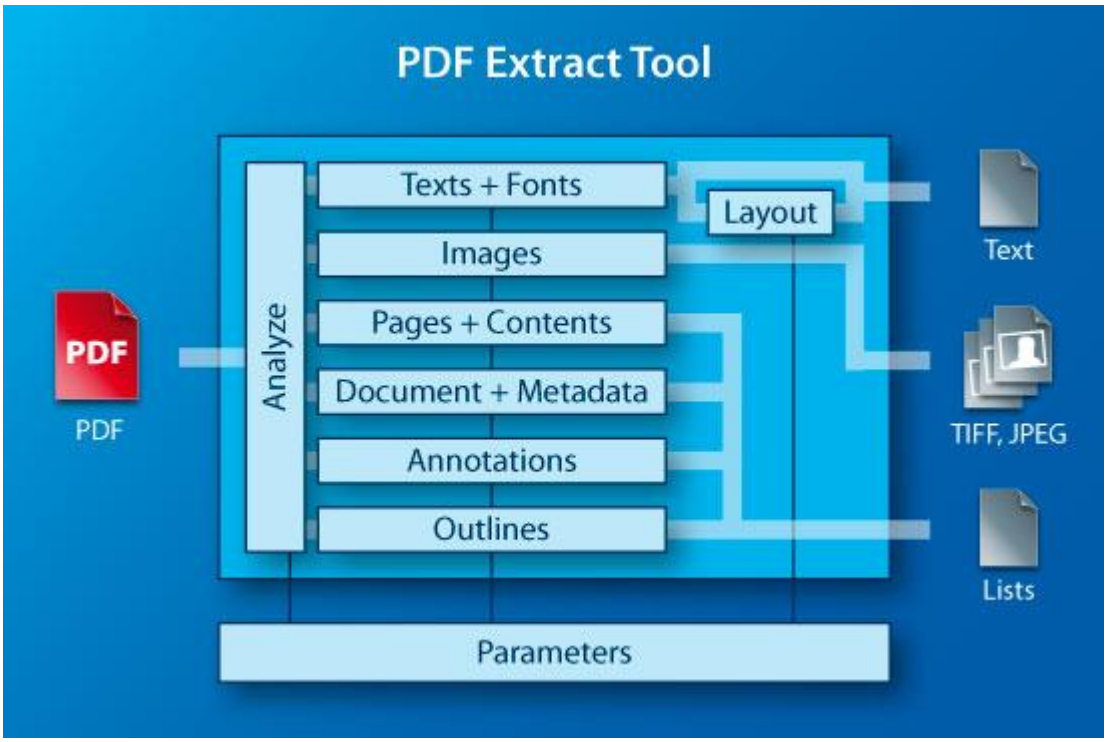
Table of Contents

1 Introduction

1.1 Description

The 3-Heights™ PDF Extract Tool is a solution for extracting and querying various attributes and page content from a PDF document. This includes texts, images, graphic objects (including paths), metadata and embedded fonts.

It is also possible to query the properties of objects. Intelligent mechanisms significantly increase extraction rates, for instance when extracting text.



1.2 Functions

The PDF Extract Tool is used to extract text, images and graphic objects (including paths) from PDF documents. Text is extractable as lines and as individual words. It is also possible to query information such as position, color, font and font size. Intelligent functions such as heuristics, word formation support and character set interpretation make it possible to restore text that is lacking essential information. The tool can also collect significant data such as position, color space and size when extracting images such as TIFF or JPEG. Querying document attributes such as PDF version, creator, author, title, subject and creation date is also possible. The tool also supports reading encrypted PDF files.

Features

- Extract text contained on a PDF page, line-wise and word-wise
- Retrieve text attributes such as position and font
- Extract graphics objects (paths)
- Extract images
- Retrieve PDF image attributes such as format, position and transparency masks
- Retrieve PDF document attributes such as page count, version number, and title
- Retrieve PDF page attributes such as the Crop Box and page rotation
- Retrieve detailed font information from PDF text
- Retrieve detailed graphics state information
- Retrieve detailed color space information
- Specify a password to decrypt PDF files

Formats

Input Formats:

- PDF 1.x (e.g. PDF 1.4, PDF 1.5)

Compliance

- Standards: ISO 32000-1 (PDF 1.7)

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)

- Installation

1.5 Installation - Software Developer Kit

The installation of the software requires the following steps.

1. Download the software, which is provided as ZIP archive from your download account.
2. Unzip the files using a tool like WinZip to a directory on your local hard disk where your program files reside. Check the appropriate option to preserve file paths (folder names). The list of files including sub-directories of the developer kit (SDK) is listed in Table: Files for Development.
3. Identify which interface (.NET, JNI, COM, C) you are using and perform the specific installation steps for that interface. These steps are described in the following chapters.

Interfaces

The 3-Heigths™ PDF Extract API provides four different interfaces. The installation and deployment of the software depend on the interface you are using.

The table below shows the supported interfaces and with which programming languages they can be used.

Table: Interfaces

Interface	*Programming Languages*
`.NET`	The MS software platform .NET can be used with any .NET capable programming language such as: • C# • VB .NET • J# • others
`JNI`	The Java native interface (JNI) is for use with Java.
`COM`	The component object model (COM) interface can be used with any COM-capable programming language, such as: • MS Visual Basic • MS Office Products such as Access or Excel (VBA) • C++ • VBScript • others
`C`	The native C interface is for use with C and C++.

Distributed Files

The software developer kit (SDK) contains all files that are used for developing the software. The roles of all files with respect to the four different interfaces is shown in Table: Files for Development. The files are split in four categories:

Req. This file is required for this interface.

Opt. This file is optional (e.g. *pdcjk.dll* is used to support Asian languages, it is not used for other languages). See also Table: File Description to identify which files are required for your application.

Doc. This file is for documentation only.

An empty field indicates this file is not used at all for this particular interface.

Table: Files for Development

Name	*.NET*	*JNI*	*COM*	*C*
`bin\PDFParser.dll`	**Req.**	**Req.**	**Req.**	**Req.**
`bin\pdcjk.dll`	Opt.	Opt.	Opt.	Opt.
`bin\*NET.dll`	**Req.**			
`bin\*NET.xml`	Doc.			
`bin\Icc\*.*`	Opt.	Opt.	Opt.	Opt.
`doc\*.pdf`	Doc.	Doc.	Doc.	Doc.
`doc\PDFParser.idl`			Doc.	
`doc\javadoc\*.*`		Doc.		
`include\expa_c.h`				**Req.**
`include\*.*`				Opt.
`jar\EXPA.jar`		**Req.**		
`lib\PDFParser.lib`				**Req.**
`samples\*.*`	Doc.	Doc.	Doc.	Doc.

The purpose of the most important distributed files of is described in Table: File Description.

Table: File Description

Name	*Description*
`bin\PDFParser.dll`	This is the DLL that contains the main functionality.
`bin\pdcjk.dll`	This DLL contains support for Asian languages. It is loaded from the module path.
`bin\*NET.dll`	The .NET assemblies are required when using the .NET interface. The files *bin*NET.xml* contain the corresponding XML documentation for MS Studio.

`bin\Icc\*.*`	The two color profiles "USWebCoatedSWOP.icc" and "sRGB Color Space Profile.icm" are required to transform RGB to CMYK values and vice versa when extracting colors. The color profiles must not be renamed, or they will not be found. Compatibility Note: In versions prior to 2.1.7, the color profiles has different names: "CMYK.icc" and "sRGB.icm". These old names are no longer supported.
`doc\*.*`	Various documentation.
`include\*.*`	Contains files to include in your C / C++ project.
`jar\EXPA.jar`	The Java wrapper.
`lib\PDFParser.lib`	The Object File Library needs to be linked to the C/C++ project.
`samples\*.*`	Contains sample programs in different programming languages.

Color Profiles

The 3-Heights™ PDF Extract API uses color profiles to convert sRGB to CMYK colors and vice versa. If no color profiles are available, the conversion is done algorithmically.

In order to convert using color profiles there are two files required: *Icc\CMYK.icc* and *Icc\sRGB.icm* where the directory *Icc* must be a direct sub-directory of where *PdfParser.dll* resides.

Color profiles can be downloaded from the links provided in the directory *Icc*. Download at least one CMYK color profile and sRGB profile or use copy them from your local systems. (Most systems have pre-installed color profiles available at *%systemroot%\system32\spool\drivers\color*.) Rename them to *sRGB.icm* and *CMYK.icc*.

1.6 Deployment - Runtime Kit

Distributed Files

The runtime kit (RTK) contains all files that are used for deploying the software. This is a subset of the files contained in the SDK. Which files are required (**Req.**), optional (Opt.) or not used (empty field) for the four different interfaces is shown in the table below.

Table: Files for Deployment

Name	*.NET*	*JNI*	*COM*	*C*
`bin\PDFParser.dll`	**Req.**	**Req.**	**Req.**	**Req.**
`bin\pdcjk.dll`	Opt.	Opt.	Opt.	Opt.
`bin\*NET.dll`	**Req.**			
`bin\Icc\*.*`	Opt.	Opt.	Opt.	Opt.

jar\EXPA.jar	**Req.**

Deploying the Application

The deployment of an application works as described below:

1. Identify the required files from your developed application
2. Identify all files from the RTK that are required by your developed application
3. Include all these files into an installation routine such as an MSI file or simple batch script
4. Perform any interface-specific actions (e.g. registering when using the COM interface)

Example

This is a very simple example of how a COM application written in Visual Basic 6 could be deployed.

1. The developed and compiled application consists of the file *TextExt.exe*.
2. The application uses the COM interface and is distributed on Windows XP only.
   - The main DLL *PDFParser.dll* must be distributed.
   - Asian text should be supported, thus *pdcjk.dll* is distributed.
3. All file are copied to the target location using a batch script. This script contains the following commands:

```
COPY TextExt.exe %targetlocation%\.
COPY PDFParser.dll %targetlocation%\.
COPY pdcjk.dll %targetlocation%\.
```

4. For COM, the main DLL needs to be registered in silent mode (/s) on the target system. This step requires PowerUser privileges and is added to the batch script.

```
REGSVR32 /s %targetlocation%\PDFParser.dll
```

1.7 Interface specific Installation Steps

COM Interface

Registration: Before you can use the 3-Heights™ PDF Extract 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 (located in *C:\windows\system32*). The following screenshot shows the registration of *PDFExtract.dll*.

If the registration process succeeds the following box is displayed:

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 *PDFParser.dll*.

Java Interface

For compilation and execution: The Java Archive *jar\EXPA.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\EXPA.jar TextExt.java
```

For execution: Additionally the Library *bin\PDFParser.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\EXPA.jar -Djava.library.path=.;C:\pdf-tools\bin TextExt input.pdf
```

.NET Interface

The 3-Heights™ PDF Extract API 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. See also chapter *"Getting Started"*.

PDFParser.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 *PDFParser.dll* is called by the .NET assembly *PdfExtractNET.dll*. *PDFParser.dll* must be found at execution time by the Windows operating system.

The common way to do this is adding *PDFParser.dll* as an existing item to the project and set its property "Copy to output directory" to "Copy if newer".

Alternatively the directory where *PDFParser.dll* resides can be added to the environment variable "PATH" or it can simply be copied manually to the output directory.

Native C Interface

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

1.8 Uninstall, Install a new version

In order to uninstall the product undo all the steps done during installation, e.g. un-register using regsvr32 –u, delete all files, etc.

Note that an expired evaluation DLL cannot be unregistered. If you would like to un-register an expired evaluation DLL, download a new (non-expired) evaluation version, overwrite the old version and un-register it.

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.

1.9 Unix

Unpack the archive in an installation directory, i.e. /User/lib/pdf-tools.

- *bin/libPDFPARSER.so*: This is the library that contains the main functionality (required)
- *doc*: Contains documentation files
- *include* : Contains files to include in your C / C++ project
- *jar/EXPA.jar*: Contains the Java wrapper

Installation on Unix Systems

1. Unpack the archive in an installation directory, e.g. */usr/pdftools.com/*
2. Copy or link the shared object into one of the standard library directories, e.g.

   ```
   ln -s /usr/pdftools.com/bin/libPDFPARSER.so /usr/lib
   ```

3. In case you have not yet installed the GNU shared libraries, get a copy of these from http://www.pdf-tools.com; extract the shared images and copy or link them into */usr/lib* or */usr/local/lib*.

Installation on Mac OS X

1. Unpack the archive in an installation directory, i.e. */User/lib/pdf-tools*
2. Add the directory containing *libPDFPARSER.dylib* to the *DYLD_LIBRARY_PATH*.

For Java

- Rename the file *libPDFPARSER.dylib* to *libPDFPARSER.jnilib* or create a file link for this purpose by using the following command:

```
ln libPDFPARSER.dylib libPDFPARSER.jnilib
```

- Add the *jar/EXPA.jar* file to the CLASSPATH.

1.10 Samples

Samples for various programming languages are included in the Windows kits. They can also be downloaded at the PDF Tools AG web site.

http://www.pdf-tools.com/asp/products.asp?name=EXPA

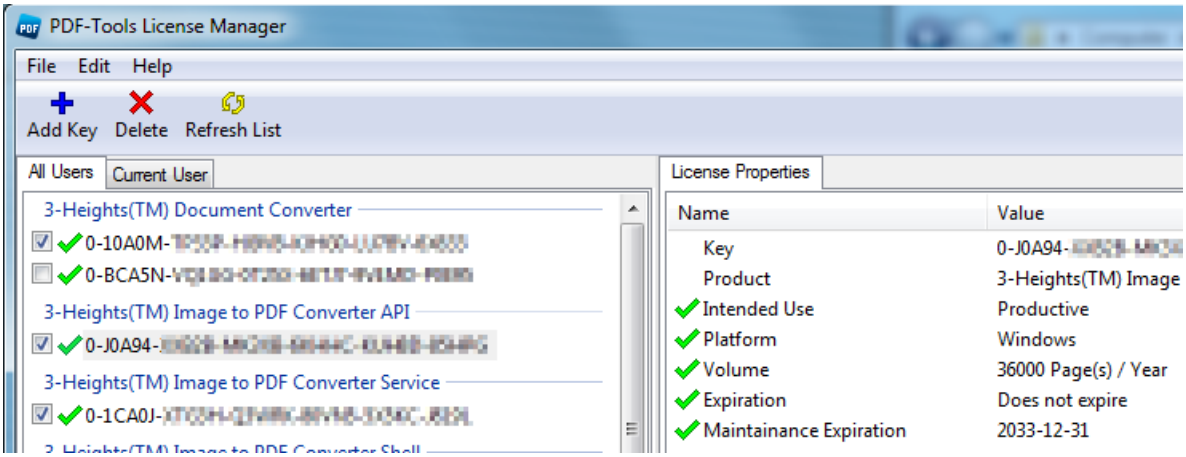
2 License Management

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.

2.1 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.

2.2 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
```

2.3 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.

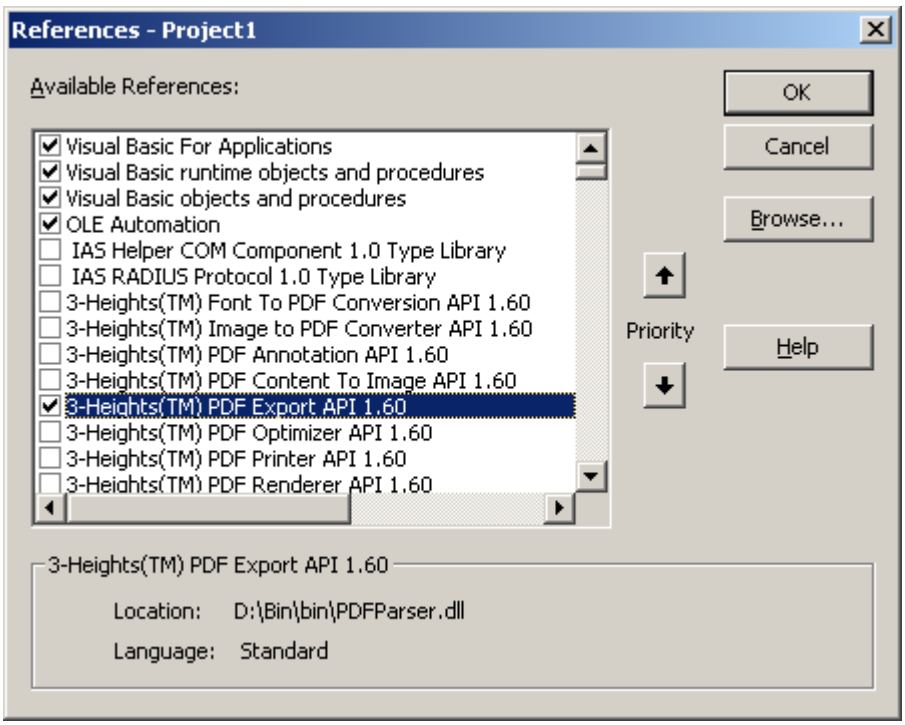
It may be necessary to change permissions to make the licenses readable for all users. Example:

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

3 Getting started

3.1 Visual Basic

In order to use the component in a Visual Basic 6 project, you have to add the component as a project reference as shown below. The version which is registered will show up.



3.2 ASP Script

The PDF Extract component can be accessed in an ASP script using the call *Server.CreateObject* and a class name as parameter. For example to create PDF Extract *Document* object, use a command like this:

set pdfDoc = Server.CreateObject("PDFParser.Document")

Here is a small ASP sample how to create a Document object and then retrieve the total number of pages in a PDF file. The path to the PDF "myfile.pdf" needs to be modified.

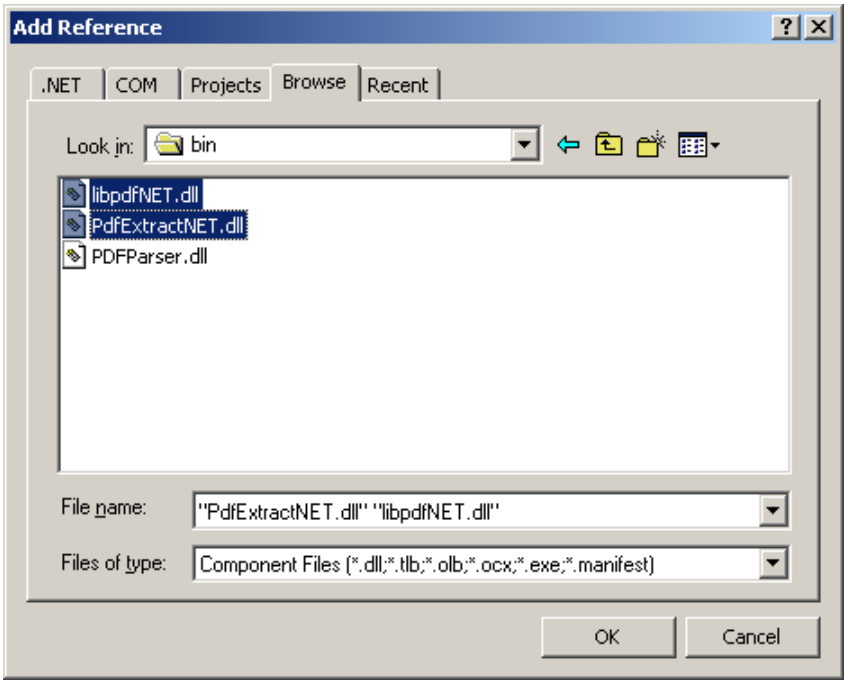
```
<%@ Language=VBScript %>
<%
      option explict
      dim pdfDoc
      set pdfDoc = Server.CreateObject("PDFParser.Document")
      if not pdfDoc.Open(Server.Mappath("myfile.pdf")) then
            Response.Write "<p>"
            Response.Write "Could not open  file." & "<br>"
      end if
      Response.Write "<p>"
      Response.Write "Number of pages: " & pdfDoc.PageCount & "<br>"
      Response.Write "</p>"
%>
```

3.3 .NET

There should be at least one .NET sample for MS Visual Studio 2005 available in the ZIP archive of the Windows Version of the 3-Heights™ PDF Extract API. Easiest for a quick start is to refer to this sample.
In order to create a new project from scratch, do the following steps:

1. Start Visual Studio and create a new C# or VB project.
2. Add a reference to the .NET assemblies.

To do so, in the "Solution Explorer" right-click your project and select "Add Reference…". The "Add Reference" dialog will appear. In the tab "Browse", browse for the .NET assemblies *libpdfNET.dll*, *RendererNET.dll* and *PdfExtractNET.dll* and add them to the project as shown below:



3. Import namespaces (Note: This step is optional, but useful.)
4. Write Code

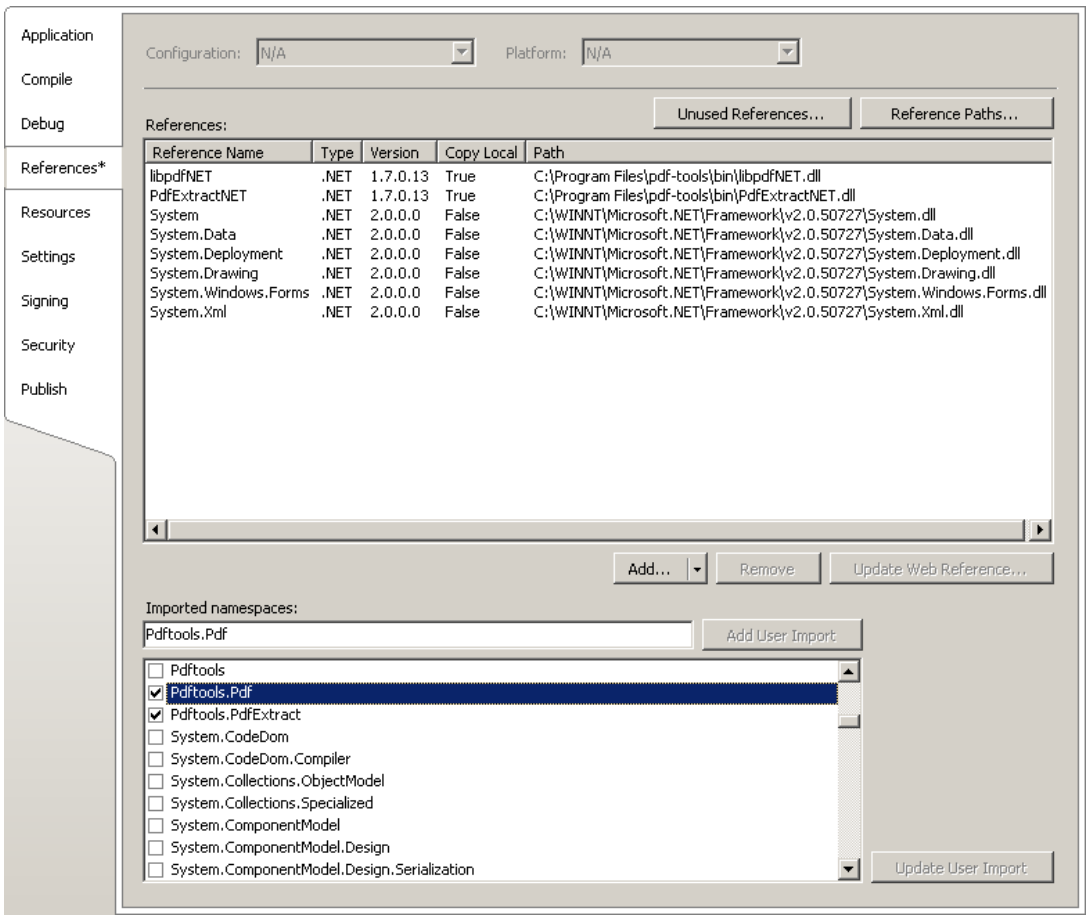
   Steps 3 and 4 are shown separately for C# and Visual Basic.

Visual Basic

3. Double-click "My Project" to view its properties. On the left hand side, select the menu "References". The .NET assemblies you added before should show up in the upper window.

   In the lower window import the namespaces *Pdftools.Pdf* and *Pdftools.PdfExtractNET*.

   You should now have settings similar as in the screenshot below:



4. The .NET interface can now be used as shown below:

```
Dim document As New Pdftools.PdfExtract.Document()
document.Open(...)
Dim content = document.Page.Content
...
```

C#

3. Add the following namespaces:

```
using Pdftools.Pdf;
using Pdftools.PdfExtract;
```

4. The .NET interface can now be used as shown below:

```
Document document = new Document();
document.Open(...);
Content content = document.Page.Content;
...
```

Trouble Shooting

The most common issue when using the .NET interface is if the native DLL is not found at execution time. This normally manifests when the constructor is called for the first time and exception is thrown - normally of type System.TypeInitializationException.

To resolve that ensure the native DLL is found at execution time. For this, see sub-chapter ".NET Interface" in the chapter "Installation".

4 Reference Manual

Note this manual describes the COM interface only. Other interfaces (C, Java, .NET) however work similarly, i.e. they have calls with similar names and the call sequence to be used is the same as with COM.

4.1 Document Interface

Author

```
Property String Author
Accessors: Get
```

Return the author from the document's info object.

Close

```
Method Void Close()
```

This method closes an open document. If the document is already closed the method does nothing.

Compliance

```
Property TPDFCompliance Compliance
```

Get the claimed compliance of the document. For instance, this property can be used in order to detect if the document claims to be PDF/A.

CreationDate

```
Property Date CreationDate
Accessors: Get
```

Return the creation date of the document's info object.

Creator

```
Property String Creator
Accessors: Get
```

Return the name of the creator of the document's info object.

GetCurrentOutlineLevel

```
Method Long GetCurrentOutlineLevel()
```

Return the level of the current outline (bookmark).

- Return value

  The level of the current outline (0 is equal to root level)

GetDestination

```
Method PDFDestination GetDestination (String Destination)
```

Return an interface to the destination specified in the parameter.

- Parameters:

  Destination: The named destination

- Return value:

  An interface to the specified destination if it exists

  Nothing otherwise

GetFirstColorSpaceResource

```
Method PDFColorSpace GetFirstColorSpaceResource()
```

Return an interface to the first color space resource (see ColorSpace Interface).

- Return value:

  An interface to the first color space resource if there is any

  Nothing otherwise

GetFirstEmbeddedFile

```
Method PDFEmbeddedFile GetFirstEmbeddedFile()
```

Return an interface to the first embedded file (see EmbeddedFile Interface). Embedded files of both the document's collection (PDF Portfolio) and of FileAttachment annotations are returned.

- Return value:

  An interface to the first embedded file if there is any

  Nothing otherwise

GetFirstFontResource

```
Method PDFFont GetFirstFontResource()
```

Return an interface to the first font resource (see Font Interface).

- Return value:

  An interface to the first font resource if there is any

  Nothing otherwise

GetFirstImageResource

```
Method PDFImage GetFirstImageResource()
```

Return an interface to the first image resource (see Image Interface).

- Return value:

  An interface to the first image resource if there is any

  Nothing otherwise

GetFirstOutlineItem

```
Method PDFOutlineItem GetFirstOutlineItem()
```

Return an interface to the first outline item (see Outline Interface).

- Return value:

  An interface to the first outline item if there is any

  Nothing otherwise

GetInfoEntry

```
Method String GetInfoEntry(String szKey)
```

Return the value of a custom entry in the info object.

- Parameters:

  szKey: The string defining the info object, such as “Author” or “Subject”.

- Return value:

  The string corresponding to the info object if it exists

  Nothing otherwise

GetNextColorSpaceResource

```
Method PDFColorSpace GetNextColorSpaceResource()
```

Return an interface to the next color space resource.

- Return value:

  An interface to the next color space resource if there is any

  Nothing otherwise

GetNextEmbeddedFile

```
Method PDFEmbeddedFile GetNextEmbeddedFile()
```

Return an interface to the next embedded file.

- Return value:

  An interface to the next embedded file if there is any

  Nothing otherwise

GetNextFontResource

```
Method PDFFont GetNextFontResource()
```

Return an interface to the next font resource.

- Return value:

  An interface to the next font resource if there is any

  Nothing otherwise

GetNextImageResource

```
Method PDFImage GetNextImageResource()
```

Return an interface to the next image resource.

- Return value:

  An interface to the next image resource if there is any

  Nothing otherwise

GetNextOutlineItem

```
Method PDFOutlineItem GetNextOutlineItem(Long MaxLevel, Boolean
ReturnOpenOnly)
```

Return an interface to the next outline item.

- Parameters:

  MaxLevel (optional, default 20): The maximum level of the depth of the outlines.

  ReturnOpenOnly (optional, default false): Return only outlines which are opened.

- Return value:

  An interface to the next outline item if there is any

  Nothing otherwise

GetObject

```
Method PDFObject GetObject(String Path)
```

This method returns a PDF object specified by the path string. The path consists of a prefix and operators.

Prefix:

- "$/": Trailer dictionary (see chapter 3.4.4 of the PDF Reference), valid entries are "$/Root", "$/Info", "$/Encrypt"
- "%n/": Page n

Path operators:

- "/name": Entry "name" of the dictionary

- "[i]": Index i in the array

Examples

- "$/Root/Pages/Kids[0]/Contents"
- "%1/Resources/Font/TT2/FontDescriptor/FontFamily"

GetOcg

```
Method Ocg GetOcg(Integer Count)
```

Return an interface to an optional content group item.

- Parameters:

  Count: The number of the optional content group. Optional content groups are numbered from 0 to OcgCount-1.

- Return value:

  An interface to an optional content group item

GetPageLabel

```
Method String GetPageLabel(Long PageNo)
```

Return the label text associated to a specific page given its number. Examples for page labels are: "7", or "vii".

- Parameters:

  PageNo: The page number

- Return value:

  A string holding the page label if a page label exists. If no page label exists the page number is converted to a string and returned.

GetXMPMetadata

```
Method Boolean GetXMPMetadata (String FileName)
```

Extract the document's XMP metadata stream and write it to the specified file.

- Parameters:

  FileName: The name of the output file

- Return value:

  True, if the document contains XMP metadata and the stream was successfully written to the output file.

GetXMPMetadataMem

```
Method Variant GetXMPMetadata ()
```

Extract the document's XMP metadata stream as a byte array. If the document does not contain XMP metadata, NULL is returned.

IsCollection

```
Property Boolean IsCollection
Accessors: Get
```

Return true if the PDF document is a collection (aka PDF Portfolio).

IsEncrypted

```
Property Boolean IsEncrypted
Accessors: Get
```

Return true if the PDF document has an encryption entry.

IsLinearized

```
Property Boolean IsLinearized
Accessors: Get
```

Return true if the linearization flag is set in the PDF document. This property does not actually validate whether the linearization is correct.

Linearization refers to optimizing the PDF for fast web access, i.e. support random page access.

Keywords

```
Property String Keywords
Accessors: Get
```

Return a string with the keywords of the document's info object.

LastError

```
Property TPDFErrorCode LastError
Accessors: Get
```

This property can be accessed to receive the latest error code. Any return value other than PDF_S_SUCCESS (0) indicates that an error occurred. See enumeration *TPDFErrorCode*.

LastErrorMessage

```
Property String LastErrorMessage
Accessors: Get
```

Return the error message text associated with the last error (see property LastError).

Note, that the property is NULL, if no message is available.

MajorVersion

```
Property Integer MajorVersion
Accessors: Get
```

Return the major version of the document. (Ex. PDF Version 1.5 corresponds to Adobe Acrobat 6, the major version is 1, the minor is 5)

MinorVersion

```
Property Integer MinorVersion
Accessors: Get
```

Return the minor version of the document.

ModDate

```
Property Date ModDate
Accessors: Get
```

Return the modification date of the info object of the document.

OcgCount

```
Property Long OcgCount
Accessors: Get
```

Get the number of optional content groups (also known as "layers") of the document.

- Return value:

  The number of optional content groups in this document

Open

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

This method opens a PDF random access disk file, i.e. makes the objects contained in the PDF document accessible. If the document is already open it is closed first.

- Parameters:

  FileName: The file name and optionally the file path, drive or server string according to the operating systems file name specification rules.

  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 was opened successfully.

  False: The file does not exists, it is corrupt, or the password is invalid.

OpenMem

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

This method opens a PDF memory block, i.e. makes the objects contained in the PDF document accessible. If the document is already open it is closed first.

- 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 was opened successfully from memory.

  False: The document in memory is not readable.

Page

```
Property PDFPage Page
Accessors: Get
```

This property allows to retrieve an interface to the currently selected page of a document.

PageCount

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

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

For collections (aka. PDF Portfolios) with no cover page, this property returns 0.

PageNo

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

This property allows to set and get the currently selected page of an open document given its page number. The numbers are counted from 1 for the first page to the value of the *PageCount* attribute for the last page. If the document is closed zero is returned.

Producer

```
Property String Producer
Accessors: Get
```

Return the name of the producer from the document's info object.

Subject

```
Property String Subject
Accessors: Get
```

Return the subject from the document's info object.

Title

```
Property String Title
Accessors: Get
```

Return the title from the document's info object.

4.2 Page Interface

ArtBox

```
Property Variant ArtBox
Accessors: Get
```

This property returns the art box rectangle given by the coordinates left, bottom, right, top. The values are returned as an array of four single precision real numbers. The art box is optional, it defines the region that contains meaningful content intended by the creator. If there is no art box set, the crop box is returned.

BleedBox

```
Property Variant BleedBox
Accessors: Get
```

Return the bleed box rectangle given by the coordinates left, bottom, right, top. The values are returned as an array of four single precision real numbers. The bleed box is optional, it defining the region to which the contents of the page should be clipped when output in a production environment. If there is no bleed box set, the crop box is returned.

Content

```
Property IPDFContent* Content
Accessors: Get
```

Return an interface to the content stream of the page (see Content Interface).

CropBox

```
Property Variant CropBox
Accessors: Get
```

Return the crop box rectangle given by the coordinates left, bottom, right, top. The values are returned as an array of four single precision real numbers. The crop box is optional, it defines the range of the visible region of the page. If there is no crop box set, the media box is returned.

DeviceColorant

```
Property String DeviceColorant
Accessors: Get
```

Return the device colorant.

Document

```
Property PDFDocument Document
Accessors: Get
```

Return the interface to the page's document (see Document interface).

GetFirstAnnotation

```
Method Annotation GetFirstAnnotation()
```

Return an interface to the first annotation (see Annotation Interface).

- Return value:

  An interface to the first annotation if any annotations exist.

  Nothing otherwise

GetNextAnnotation

```
Method Annotation GetNextAnnotation()
```

Return an interface to the next annotation.

- Return value:

  An interface to the next annotation if any further annotations exist.

  Nothing otherwise.

MediaBox

```
Property Variant MediaBox
Accessors: Get
```

Return the media box rectangle given by the coordinates left, bottom, right, top. The values are returned as an array of four single precision real numbers. The media box is required, it defines the physical boundaries of the medium on which the page is intended to be displayed or printed.

Rotate

```
Property Integer Rotate
Accessors: Get
```

Return the rotation value of the page. This value is used by viewer programs to turn the page by the given number of degrees while displaying. 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, 270.

TrimBox

```
Property Variant TrimBox
Accessors: Get
```

Return the trim box rectangle given by the coordinates left, bottom, right, top. The values are returned as an array of four single precision real numbers. The trim box is optional, it defines the intended dimensions of the finished page after trimming. If there is no trim box set, the crop box is returned.

4.3 Content Interface

BreakWords

```
Property Boolean BreakWords
Accessors: Get, Set
Default: True
```

This property is deprecated and superseded by the TextExtConfiguration property. In order to get the same behavior as with BreakWords, use the following options:

- BreakWords true: Set the eTECBreakSpaceUnicode flag and clear the flags eTECPosMergeSingleSpace and eTECPosMergeMultiSpace.
- BreakWords false: Clear the eTECBreakSpaceUnicode flag and set the flags eTECPosMergeSingleSpace and eTECPosMergeMultiSpace.

BoundingBox

```
Property Variant BoundingBox
Accessors: Get, Set
Default: CropBox of the page
```

The bounding box is a rectangle in user space units (1/72 inch). The rectangle is used, when the Reset() method is called with AccountForRotate set to TRUE and has an effect on the coordinate transform. The bounding box must be set before calling Reset().

ExpandLigatures

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

When *ExpandLigatures* is set to true, ligatures such as fi, ff, fl, etc. found during text extraction are converted to individual characters.

Flags

```
Property Long Flags
Accessors: Get
```

Return –1 while content is parsed and the annotation flags when annotations are parsed. (see also Property *Flags* in the Annotation interface)

GetNextImage

```
Method PDFImage GetNextImage()
```

This method reads the content stream objects until an image object can be returned or the end of the content stream is reached. If an image object could be found, an interface to the image object (see Image Interface) is returned. Its interface can also be retrieved through the content's Image property. The graphics state can be retrieved through the content's GraphicsState property.

- Return value:

  An interface to the next image object on the current page if there is any.

  Nothing otherwise.

GetNextObject

```
Method TPDFContentObject GetNextObject()
```

This method reads the content stream objects until a text, image, or path object can be returned or the end of the content stream is reached.

- Return values:

  eNone: The end of the content stream has been reached and the content's *Path* property doesn't return a valid value.

  eText: A text object could be composed and its interface can be retrieved through the content's *Text* property.

  eImage: An image object could be found and its interface can be retrieved through the content's Image property. The graphics state can be retrieved through the content's *GraphicsState* property.

  ePath: A path object could be found and its string representation can be retrieved through the content's Path property. The graphics state can be retrieved through the content's *GraphicsState* property.

  eSave: Save the current graphics state on the graphics state stack.

eRestore: Restore the graphics state by removing the most recently saved state from the stack and making it the current graphics state.

eBeginOCM: Start of a sequence of objects, whose visibility is defined by an optional content membership string (property OCM). Sets the property OCM. OCM sequences can be nested.

eEndOCM: Marks the end of an OCM sequence.

GetNextPath

```
Method String GetNextPath()
```

This method reads the content stream objects until a path object can be returned or the end of the content stream is reached. If a path object could be found, a string representation of a path object is returned. It can also be retrieved through the content's Path property. The graphics state can be retrieved through the content's GraphicsState property.

- Return value:

  The next text path on this page if there is any.

  Nothing otherwise.

GetNextText

```
Method PDFText GetNextText()
```

This method reads the content stream objects until a text object can be returned or the end of the content stream is reached. If a text object can be found, an interface to the next read text object (see Text Interface) is returned. In contrast to the methods *GetNextImage* and *GetNextPath* this method reads text objects and merges text objects until a major text property (font, line coordinate, etc.) changes or a word break occurs if word breaking is enabled (see Property *BreakWords*). The current graphic state can be retrieved through the current content object's interface.

- Return value:

  An interface to the next text object if there is any one this page.

  Nothing otherwise.

GraphicsState

```
Property TPDFGraphicsState GraphicsState
Accessors: Get
```

Return an interface to the content's graphics state (see GraphicsState Interface). The graphics state is updated each time a method *GetNextText*, *GetNextImage*, *GetNextPath*, or *GetNextObject* is called.

IgnoreOCM

```
Property Boolean IgnoreOCM
```

```
Accessors: Get, Set
```

Option to ignore optional content membership and make all content visible. BeginOCM and EndOCM objects are extracted, but they have no effect on the extracted content. E.g. when true, hidden text is extracted as well. Set this property to true in order to extract all content.

Image

```
Property PDFImage Image

Accessors: Get
```

Return an interface to the last read image object (see Image Interface). The image object is updated each time the method *GetNextImage* or *GetNextObject* is called.

OCM

```
Property String OCM

Accessors: Get
```

Return the current optional content membership string which defines the visibility as Boolean function of OCG in C syntax. OCGs are represented by Ids. Retrieve the respective OCG using the Document interface's GetOcg method.

supported operators: "&&", "||", "!"

Example: "1 && 2" means, that the following objects are visible only, if OCG 1 and OCG 2 are visible

Note: This property is valid only immediately after extraction of BeginOCM object.

Path

```
Property String Path

Accessors: Get
```

Return the last read path object in its string form. The path object describes a graphic drawing consisting of stroked lines and curves as well as filled shapes. The string contains the PDF path construction tokens consisting of real value operands (in angle brackets) followed by operator mnemonics:

- Move current point to: <x> <y> m
- Line from current point to: <x> <y> l
- Rectangle: <x> <y> <w> <h> re
- Cubic Bezier curve from current point to: <x1> <y1> <x2> <y2> <x3> <y3> c
- Close figure (move to start of last sub-path): h
- Fill path: f
- Stroke path: s
- End path (without filling and stroking): n
- Modify current clipping path: W

The exact details to the path construction operators can be found in Adobe's PDF Reference Manual. The path object is updated each time the method *GetNextPath* or *GetNextObject* is called. This property cannot be set.

Reset

```
Method Void Reset(Boolean AccountForRotate)
```

This method allows to reset the content extraction process and set the point of extraction to the beginning of the content stream.

- Parameters:

  AccountForRotate (Optional, default=false): This property defines origin and orientation of the coordinate system of the coordinates of extracted content elements. The unit of the coordinate system is 1/72 inch.

  - *False*: The coordinates are extracted as raw coordinates as used in the PDF document.
  - *True*: Extracted coordinates are relative to the bottom left corner of the visible page as displayed by a viewer. I.e. the page is rotated by the page's Rotate attribute and cropped using a bounding box. For example, the coordinate (0, 0) denotes the bottom left corner of the page.
    The default bounding box used is the CropBox. This can be changed by setting the BoundingBox property before calling the Reset method.

SpaceFactor

```
Property Single SpaceFactor
Accessors: Get, Set
```

This property can be used to get or set the distance between two characters that is required to insert a blank for text extraction. The default is 0.3. This means any distance between two characters that are further apart as 0.3 times the width of the space character glyph in this font is interpreted as a new word. For text that is written very narrowly, this property should be decreased in order to avoid concatenation of words.

Text

```
Property PDFText Text
Accessors: Get
```

Return an interface to the last read text object (see Text Interface). The text object is updated each time the method *GetNextText* or *GetNextObject* is called.

TextExtConfiguration

```
Property Long TextExtConfiguration
Accessors: Get, Set
```

```
Default: 7 (eTECBreakTextState + eTECBreakGraphicsState +
eTECBreakSpaceUnicode)
```

This property serves to control the way the text extraction algorithm works. Text extraction collects all text objects and merges them into a single text. This property controls which text objects are merged. See the Enumeration TPDFTextExtractConfiguration for a list of all possible options.

Recommended settings for different use cases:

- *Text search or indexing*
  i.e. text formatting is not important
  - Extract Words individually: *eTECBreakSpaceUnicode*
  - Extract phrases: *eTECPosMergeSingleSpace* + *eTECPosMergeMultiSpace*
- *Conversion of pdf content to another format*
  i.e. text formatting and exact positioning is crucial
  - Usage of RawString or extracted fonts: *eTECBreakTextState* + *eTECBreakGraphicsState*
  - Other: *eTECBreakTextState* + *eTECBreakGraphicsState* + *eTECPosMergeSingleSpace*

TranslateSymbolic

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

Replace symbolic character from the Unicode custom range (0xF000..0xF0FF) with WinAnsi codes (0x00..0xFF).

4.4 Image Interface

Alternates

```
Property Variant Alternates
Accessors: Get
```

Return an array of alternate images (see Interface AlternateImage). An image can have none, one or multiple alternate images.

BitsPerComponent

```
Property Integer BitsPerComponent
Accessors: Get
```

Return the number of bits that are used to represent a single color component of an image sample. The number of color components per image data sample can be retrieved through the image's color space interface.

ChangeOrientation

```
Method Boolean ChangeOrientation(TPDFOrientation Orientation)
```

Set the orientation of the image. This value has to be set prior to using the method *Store()*. The orientation of the image can be retrieved from the property *GraphicsState.ctm.Orientation*.

ColorSpace

```
Property IPDFColorSpace* ColorSpace
Accessors: Get
```

Return an interface to the color space of the image (see ColorSpace Interface).

Compression

```
Property IPDFCompression Compression
Accessors: Get
```

Return the compression used for the image in the pdf.

ConvertToRGB

```
Method Boolean ConvertToRGB()
```

Convert the image to an RGB image. The conversion uses the image's color space to interpret the sample data. Calibrated color spaces are converted to RGB values according to the sRGB color standard. Device color space are converted using pre-defined color profiles.

- Return value:

  True if the conversion was successful.

  False otherwise.

GetImage

```
Method Variant GetImage()
```

Return the image from memory which was previously saves using the method *StoreInMemory*.

- Return value:

  The image as a 1-dimensional byte array.

GetResolution

```
Method Single GetResolution(IPDFTransformMatrix* Matrix)
```

Return the resolution of an image on the page in dpi (dots per inch).

- Parameters:

Matrix: The transformation matrix of the image. This parameter is required since the image itself has no resolution. The resolution is the ratio between the size of the image and the size it uses on the page.

- Return values:

  The calculated resolution in dpi.

Height

```
Property Long Height
Accessors: Get
```

Return the height of the image in pixels (also called samples). The unit of pixels can be converted to a distance unit such as inch, millimeter etc. using a resolution value, i.e. 72 dpi (dots per inch).

IsBitonal

```
Property Boolean IsBitonal
Accessors: Get
```

Return true when the image is bi-tonal.

IsColor

```
Property Boolean IsColor
Accessors: Get
```

Return true when the image is color.

IsMonochrome

```
Property Boolean IsMonochrome
Accessors: Get
```

Return true when the image is monochrome.

ObjNumber

```
Property Long ObjNumber
Accessors: Get
```

Returns a unique number of this image resource. If the number is 0, the image resource occurs once only in the document (i.e. it is an inline image). If the number is larger than 0, the image resource might be used multiple times.

IsMonochrome

```
Property Boolean IsMonochrome
Accessors: Get
```

Return true when the image is monochrome.

Samples

```
Property Variant Samples
Accessors: Get
```

Return the image's data samples in a byte array. The sample data is ordered by line from top to bottom and within a line from left to right. The lines are byte aligned. If the number of bits per component is less than one byte then the samples are ordered beginning with the most significant bit first.

If the property *ImageMask* of the image is set to False, the interpretation of the sample data must be done according to the properties in the color space of the image.

If the property *ImageMask* of the image is set to True, the sample data represents a stencil mask. In this case the color space isn't meaningful and the data is organized one bit per pixel. A one bit signifies a transparent pixel and a zero bit signifies a pixel with the current fill color (see GraphicsState Interface).

SMask

```
Propertiy Variant SMask
```

With this property the soft mask of an image can be extracted.

Store

```
Method Boolean Store(String FileName, TPDFCompression Compression)
```

Store the image as a file.

- Parameters:

  FileName: The name of the disk file include path, drive, or Server string according to the operating system's naming rules. The type of the image is defined by its extension (".jpg" or ".tif").

  Compression (optional): The compression type (for TIFF images). The default value is *eComprDefault*.

- Return values:

  True: The file has successfully been written.

  False: An error has occurred and the disk file is unusable.

StoreInMemory

```
Method Boolean StoreInMemory(String Extension, TPDFCompression
Compression)
```

Store the image in memory. The saved image can be retrieved using the method GetImage.

- Parameters:

Extension: The type of the image is defined by its extension (".jpg" or ".tif").

Compression (optional): The compression type (for TIFF images). The default value is *eComprDefault*.

- Return values:

  True: The image has successfully been saved.

  False otherwise.

Width

```
Property Long Width

Accessors: Get
```

Return the width of the image in pixels (also called samples). The unit of pixels can be converted to a distance unit such as inch, millimeter etc. using a resolution value, i.e. 72 dpi (dots per inch).

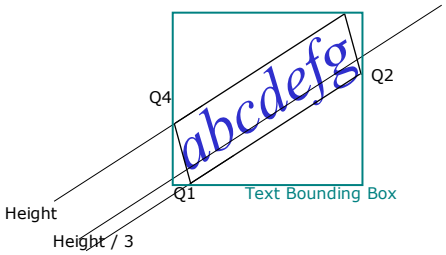
4.5 Text Interface

BoundingBox

```
Property Variant BoundingBox

Accessors: Get
```

Return the smallest rectangle that encloses the text as shown below:



The text bounding box is a rectangle which encloses the four points Q1, Q2, Q3, Q4. The points Q1 and Q2 are 1/3 of the height below the baseline.

The text bounding box is defined by four values which represent the coordinate of the lower left and the upper right corner.

FontSize

```
Property Single FontSize

Accessors: Get
```

Return the size of the font in points. The size can also be interpreted as the height of the text.

Length

Deprecated, use *StringLength* instead.

RawString

```
Property Variant RawString
Accessors: Get
```

For simple fonts this property returns the raw character codes from the PDF as a byte array. For CID fonts this property is NULL. If the *ExpandLigatures* property is not set, the length of the *RawString* is the same as the length of the *UnicodeString* and the character position vector applies to the *RawString* character codes as well.

The property *UnicodeString* always returns a string of Unicodes. These Unicodes are the result of the mapping of character codes to Unicodes defined by the PDF specification and our set of heuristics. These Unicodes might not be accurate. In some cases, you might have prior knowledge about this specific font and know the mapping of character codes to Unicodes yourself. E.g. you know the creator used the EBCDIC encoding. For this reason, the property *RawString* returns the string of character codes and allows you to apply your own mapping.

With RawString, do not use the TextExtConfiguration options *eTECBreakSpaceUnicode*, *eTECPosMergeSingleSpace* and *eTECPosMergeMultiSpace*, because the Unicode these options work with, might not be accurate.

Rotation

```
Property Single Rotation
Accessors: Get
```

Return the rotation of the string in radians (rad). (2 pi rad = 360°)

StringLength

```
Property Integer StringLength
Accessors: Get
```

Return the number of characters in the string.

UnicodeString

```
Property String UnicodeString
Accessors: Get
```

Return the text as a Unicode UTF-16 encoded string. The number of bytes per character is a multiple of two. For most languages such as English a character can be mapped to a single 16-Bit Unicode value. Complex languages such as Chinese can return multiple 16-Bit values per character. Some text strings, however, cannot be correctly mapped or cannot be mapped at all. The former is the case if e.g. the PDF creator program didn't use correct names for the character in the font encoding (see Font Interface). The latter is the case if e.g. the PDF creator program didn't embed Unicode mapping information for a symbolic font.

Width

```
Property Single Width
Accessors: Get
```

Return the width of the string in points.

XPos, YPos

```
Property Variant XPos
Property Variant YPos
Accessors: Get
```

Return the X and Y position of the characters. The return value is a 1-dimensional array holding the positions of all characters.

If the a Text contains n characters:

XPos[0] represents the 1st character

XPos[n-1] represents the last character.

XPos[n] is a calculated, virtual position of where the next character *would* start. This position and the actual position of the next character can be compared to decide whether they belong to the same word, or not.

4.6 GraphicsState Interface

Entries which have a complex structure, such as a function, are not retrievable with the 3-Heights™ PDF Extract Tool. These are for example "black generation functions" BG, "transfer functions" TR or "under-color-removal functions" UCR.

The extract tool has the ability to return colors in RGB or CMYK. If the requested color space is different from the actual color space in the PDF, the color conversion is down using color profiles.

AlphaIsShape

```
Property Boolean AlphaIsShape
Accessors: Get
```

Return the 'AlphaIsShape' flag. It is true if the soft mask contains shape values, it returns false for opacity.

BlendMode

```
Property String BlendMode
Accessors: Get
```

Return the name of the blend mode. A blend mode can be "Normal", "Multiply", "Screen", "Overlay", etc.

CharSpacing

```
Property Single CharSpacing
Accessors: Get
```

Return the current space between two characters of a text string as a single precision real number in text units.

CTM

```
Property PDFTransformMatrix CTM
Accessors: Get
```

Return an interface to the current transform matrix. The transform describes the transformation of the graphic object's coordinates from user units to page units including the effect of the page rotate attribute if requested (see method *Reset* of the Content Interface).

DashArray

```
Property Variant DashArray
Accessors: Get
```

Return the dash array of a line dash pattern. The line dash pattern controls the pattern of dashes and gaps used to stroke paths.

DashPhase

```
Property Single DashPhase
Accessors: Get
```

Return the dash phase of a line dash pattern. The dash phase is the offset of the pattern and can be larger as the pattern itself.

FillAlphaConstant

```
Property Single FillAlphaConstant
Accessors: Get
```

Return the alpha constant for filling.

FillColorCMYK

```
Property Long FillColorCMYK
Accessors: Get
```

Return the CMYK color quad for filling operations. The color value is obtained by converting the color values of the property *FillColor* by means of the FillColorSpace. The CMYK quads are encoded using the following formula: Quad = (((C * 256) + M) * 256 + Y) * 256 + K.

If a color doesn't exist (e.g. with an uncolored pattern) then –1 is now returned.

Hexadecimal:

Quad = 0xCCMMYYKK, where CC is the byte for the cyan value in the range from 0x00 to 0xFF, MM is magenta, YY is yellow, KK is key (black).

Decimal:

To retrieve the values for cyan, magenta, yellow and key apply the following formulas (VB code taking into account negative values, using integer-division \ and bitwise and **And**):

```
Quad = PDFPARSERLib.GraphicsState.FillColorCMYK
t = Quad And &H7FFFFFFF
C = t \ 16777216
M = (t \ 65536) And 255
Y = (t \ 256) And 255
K = t And 255
If Quad < 0 Then C = C Or &H80
```

There are also other ways to retrieve these values than using the above formulas.

FillColorRGB

```
Property Long FillColorRGB
Accessors: Get
```

Return the RGB color triple for filling operations. The color value is obtained by converting the color values of the property *FillColor* by means of the FillColorSpace. The RGB triples are encoded using the following formula: Triple = ((B * 256) + G) * 256 + R.

If a color does not exist (e. g. with an uncolored pattern) then –1 is now returned.

Hexadecimal:

Triple = 0xBBGGRR, where BB is the byte for the blue value in the range from 0x00 to 0xFF, GG is green, RR is red.

Decimal:

To retrieve the values for blue, green, red, apply the following formulas (integer-division \ and bitwise and **And**):

```
Triple = PDFPARSERLib.GraphicsState.FillColorRGB
B = Triple \ 65536
G = (Triple \ 256) And 255
R = Triple And 255
```

Example:

```
Triple = 8388736    (purple)
B = 8388736 \ 65536 = 128
G = (8388736 \ 256) And 255 = 0
R = 8388736 And 255 = 128
```

There are also other ways to retrieve these values than using the above formulas.

FillColorSpace

```
Property PDFColorSpace FillColorSpace

Accessors: Get
```

Return an interface to the current color space that is used for filling operations (see ColorSpace Interface). The color space is used to interpret color values of the property *FillColor*.

FillOverprintFlag

```
Property Boolean FillOverprintFlag

Accessors: Get
```

Return the overprint flag for painting operations other than stroking.

FlatnessTolerance

```
Property Single FlatnessTolerance

Accessors: Get
```

Return the flatness tolerance. Must be a positive number. A small number means higher precision.

Font

```
Property IPDFFont* Font

Accessors: Get
```

Return an interface to the text's font object that describe the character encoding as well as the shape of the character glyphs.

FontSize

```
Property Single FontSize

Accessors: Get
```

Return the current font size for text strings as a single precision real number in text units. It doesn't include any scaling factors from coordinate transforms such as from the current transform matrix or the text matrix. In order to obtain the font size in page units the values of the current text matrix have to be examined.

HorizontalScaling

```
Property Single HorizontalScaling

Accessors: Get
```

Return the current horizontal scaling factor that describes the amount of horizontal stretching of a text string. A value of greater than 1.0 stretches the string whereas a value of less than 1.0 lets the string appear as condensed.

Leading

```
Property Single Leading

Accessors: Get
```

Return the current leading (line spacing) of a text string as a single precision number in text units.

LineCap

```
Property Integer LineCap

Accessors: Get
```

Return the line cap style. The line cap style specifies the shape to be used at the end of open sub-paths and dashes when they are stroked.

0 Butt cap
1 Round cap
2 Projecting square cap

LineJoin

```
Property Integer LineJoin

Accessors: Get
```

This property returns the line join style. The line join style specifies the shape to be used at the corners of paths that are stroked.

0 Miter join
1 Round join
2 Bevel join

LineWidth

```
Property Single LineWidth

Accessors: Get
```

Return a single precision real number in user units of the line width.

MiterLimit

```
Property Single MiterLimit

Accessors: Get
```

Return the miter limit. The miter limit imposes a maximum on the ratio of the miter length to the line width, which can be fairly large when two line segments meet at a sharp angle. When the limit is exceeded, the join is converted from a miter to a bevel.

OverprintMode

```
Property Integer OverprintMode
```

Return the overprint mode.

RenderingIntent

```
Property String RenderingIntent
```

Return the name of the rendering intent.

SmoothnessTolerance

```
Property Single SmoothnessTolerance
Accessors: Get
```

Return the smoothness tolerance. The values are in the range [0.0, 1.0] where 1.0 corresponds to 100%.

SoftMask

```
Property IPDFImage* SoftMask
Accessors: Get
```

Return the soft mask as image.

StrokeAdjustment

```
Property Boolean StrokeAdjustment
Accessors: Get
```

Return the flag for the automatic stroke adjustment.

SpaceWidth

```
Property Float SpaceWidth
Accessors: Get
```

Get the width of the space character in text space. To get page user units transform using the text's matrix. The *SpaceWidth* property can be used to implement your own word breaking algorithm. For more information about this, read the descriptions of the properties *BreakWords* and *SpaceFactor*.

StrokeAlphaConstant

```
Property Single StrokeAlphaConstant
Accessors: Get
```

Return the current alpha stroke constant.

StrokeColorCMYK

```
Property Long StrokeColorCMYK

Accessors: Get
```

Return the CMYK color quad for stroking operations. The color value is obtained by converting the color values of the property *StrokeColor* by means of the StrokeColorSpace. The CMYK quads are encoded using the following formula: Quad = (((C * 256) + M) * 256 + Y) * 256 + K.

StrokeColorRGB

```
Property Long StrokeColorRGB

Accessors: Get
```

Return the RGB color triple for stroking operations. The color value is obtained by converting the color values of the property *StrokeColor* by means of the StrokeColorSpace. The RGB triples are encoded using the following formula: Triple = ((R * 256) + G) * 256 + B.

StrokeColorSpace

```
Property PDFColorSpace StrokeColorSpace

Accessors: Get
```

Return an interface to the current color space that is used for stroking operations (see ColorSpace Interface). The color space is used to interpret color values of the property *StrokeColor*.

StrokeOverprintFlag

```
Property Boolean StrokeOverprintFlag

Accessors: Get
```

This property returns the overprint flag for stroking painting operations.

TextKnockout

```
Property Boolean TextKnockout

Accessors: Get
```

Return the text knockout flag. This Boolean flag determines what text elements are considered elementary objects for purposes of color compositing in the transparent imaging model.

TextRenderingMode

```
Property Short TextRenderingMode

Accessors: Get
```

Return a value that indicates whether the text should be stroked, filled, used as a clip path or some combination of the three. The meaning of the values in detail is:

0 Fill text.
1 Stroke text.
2 Fill, then stroke text.
3 Neither fill nor stroke text (invisible).
4 Fill text and add path for clipping.
5 Stroke text and add path for clipping.
6 Fill, then stroke text and add path for clipping.
7 Add path for clipping.

TextRise

```
Property Single TextRise

Accessors: Get
```

Return a single precision real number in un-scaled text units that indicates by which amount the base line of the text is moved up or down. It is most commonly used to display subscripts and superscripts.

WordSpacing

```
Property Single WordSpacing

Accessors: Get
```

Return the current space between two words of a text string as a single precision real number in text units.

(For further information about the Graphic State, see PDF Reference, chapter 4.3.)

4.7 Font Interface

Ascent

```
Property Single Ascent

Accessors: Get
```

Return the Ascent value. This value represents the maximum height above the baseline reached by the glyphs in the font, excluding the height of glyphs for accented characters.

AvgWidth

```
Property Single AvgWidth

Accessors: Get
```

Return the average width of the glyphs in the font.

BaseName

```
Property String BaseName
Accessors: Get
```

Return the font name.

CapHeight

```
Property Single CapHeight
Accessors: Get
```

Return the height of the top of flat capital letters, measured from the baseline.

Charset

```
Property String Charset
Accessors: Get
```

Return a string listing the character names defined in a font subset. This property is only useful for Type1 fonts.

Descent

```
Property Single Descent
Accessors: Get
```

Return the Descent value. This negative number represents the maximum depth below the baseline reached by the glyphs in the font.

Encoding

```
Property Variant Encoding
Accessors: Get
```

Return the glyph name of each character.

Flags

```
Property Long Flags
Accessors: Get
```

Return the flags of the font. The flags are listed the following table. Bit positions within the flag word are numbered from 1 (low-order) to 32 (high-order).

Bit Position	Name	Meaning
1	FixedPitch	All glyphs have the same width.
2	Serif	Glyphs have serifs.
3	Symbolic	The font contains characters outside the standard Latin character set.

4	Script	Glyphs resemble cursive handwriting.
6	NonSymbolic	Font uses standard Latin character set or a subset of it.
7	Italic	Glyphs are italic.
17	AllCap	Font has no lowercase letters.
18	SmallCap	Lowercase letters are small uppercase letters.
19	ForceBold	If set, bold glyphs are painted bold even at very small text size.

FontBBox

```
Property Variant FontBBox

Accessors: Get
```

Return the font bounding box. The font bounding box is the rectangle in which all glyphs would fit, if they were placed on top of each other with their origins at the same point.

FontFile

```
Property Variant FontFile

Accessors: Get
```

Return a stream that contains a Type1 font program.

FontFileType

```
Property Integer FontFileType

Accessors: Get
```

Return the type of the font. A value of 1 corresponds to a Type 1 font program. A FontFile2 contains a TrueType font program. In most cases a value of 1, 2 or 3 will be returned.

ItalicAngle

```
Property Single ItalicAngle

Accessors: Get
```

Return the counter-clockwise angle of the dominant vertical strokes of the font.

Leading

```
Property Single Leading

Accessors: Get
```

Return the desired spacing between baselines of consecutive lines of text.

MaxWidth

```
Property Single MaxWidth
Accessors: Get
```

Return the maximum width of the glyphs in the font.

MissingWidth

```
Property Single MissingWidth
Accessors: Get
```

Return the value of the width which is used for character codes for which the glyph is missing in the font directory's Width array.

StemH, StemV

```
Property Single StemH
Property Single StemV
Accessors: Get
```

These properties return the vertical and horizontal thickness of the dominant vertical and horizontal stems of the glyphs in the font.

Type

```
Property Single Type
Accessors: Get
```

Return the font type as string.

Widths

```
Property Variant Widths
Accessors: Get
```

Return an array which contains the widths of the glyphs.

XHeight

```
Property Single XHeight
Accessors: Get
```

Return the maximum height of flat non-ascending lowercase letters (such as the letter x) measured from the baseline.

(For further information about font descriptors, see PDF Reference, chapter 5.7.)

4.8 ColorSpace Interface

BaseColorSpace

```
Property IPDFColorSpace* BaseColorSpace
Accessors: Get
```

Return a IPDFColorSpace interface to the base color space if it is existing.

ColorantName

```
Property Variant ColorantName
Accessors: Get
```

Return the name of the colorant.

Interface Note:

COM: A variant containing an array of strings is returned. These strings represent the name of the colorants of the color space. In an RGB color space these are "Red", "Green", "Blue".

C, .Net: An additional parameter is passed which defines the index of the colorant. Instead of a array containing all strings a single string is returned, e.g. "Red".

ComponentsPerPixel

```
Property Integer ComponentsPerPixel
Accessors: Get
```

Return the number of components per pixel.

HighIndex

```
Property Integer HighIndex
Accessors: Get
```

Return the highest value of the indexed colors. It is 0 when no indexed color space is used.

IsColor

```
Property Boolean IsColor
Accessors: Get
```

Return true when the color space is color.

IsIndexed

```
Property Boolean IsIndexed
Accessors: Get
```

Return true when the image uses indexed colors.

IsMonochrome

```
Property Boolean IsMonochrome
Accessors: Get
```

Return true when the color space is monochrome.

Lookup

```
Property Variant Lookup
Accessors: Get
```

Return the lookup table.

Name

```
Property String Name
Accessors: Get
```

Return the name of the color space as string (for example "DeviceGrey", "DeviceRGB" or "Indexed").

4.9 TransformMatrix Interface

a, b, c, d, e, f

```
Property Single a
Property Single b
Property Single c
Property Single d
Property Single e
Property Single f
Accessors: Get
```

The transformation matrix in PDF is specified by six numbers. All information about orientation, rotation, scaling, skewing and translation can be calculated based on these six numbers. However PDF Extract also provides properties which compute these values.

The values e and f represent the translation. In a matrix [1 0 0 1 e f], *e* is the distance on the x-axis from the left side page border, *f* is the distance on the y-axis from the bottom. (0,0) is in the lower left corner, on an page with a size of A4 portrait, (595,842) is in the upper right corner.

The scale factor in a matrix [a 0 0 d 0 0] can be obtained from the values *a* and *d* for x and y scaling respectively. With respect to fonts, *d* represents the font size of horizontal text.

A rotation of the axis by an angle α counter clockwise is produced by a matrix [cos α sin α -sin α cos α 0 0].

More detailed information can be found in the PDF Reference manual chapter 4.2.2.

Orientation

```
Property TPDFOrientation Orientation

Accessors: Get
```

Return the orientation rounded to the next 90 degrees. The orientation is an enumeration with eight different values (rotation times flipping). See enumeration *TPDFOrientation*.

Rotation

```
Property Single Rotation

Accessors: Get
```

Return the rotation angle of the matrix counter clockwise. This is equal to the minimum of XSkew and –YSkew.

XScaling, YScaling

```
Property Single XScaling

Property Single YScaling

Accessors: Get
```

Return the x and y scaling factor.

XSkew, YSkew

```
Property Single XSkew

Property Single Yskew

Accessors: Get
```

Return the x and y-axis skewing. The transformation matrix [1 tan α tan β1 0 0] skews the x-axis by α and the y-axis by β. Skewing sometimes is used to transform a regular font to italic.

XTranslation, YTranslation

```
Property Single XTranslation

Property Single Ytranslation

Accessors: Get
```

Return the X and Y translation. These are the same values as returned by the properties *e* and *f*.

4.10 Alternate Image Interface

DefaultForPrinting

```
Property Boolean DefaultForPrinting
Accessors: Get
```

Return true if the alternate image is set as default for printing.

Image

```
Property IPDFImage* Image
Accessors: Get
```

Return an interface to the alternate image (see Image Interface).

4.11 Annotation Interface

AttachedFile

```
Property IPDFEmbeddedFile AttachedFile
Accessors: Get
```

Return the embedded file attached to this annotation. This property is meaningful for FileAttachment annotations only. Note that the *AttachedFile* might not have an embedded file stream, but reference an external file via the *FileName* property only.

Color

```
Property Long Color
Accessors: Get
```

Return to color of the annotation.

Contents

```
Property String Contents
Accessors: Get
```

Return the content of the annotation.

Date

```
Property Date Date
Accessors: Get
```

Return the date of the annotation.

The used format is: #dd.mm.yyyy hh:mm:ss#

Dest

```
Property IPDFDestination* Dest
Accessors: Get
```

Return the destination of a link annotation. This entry is permitted if an A (action) entry is present.

Flags

```
Property Long Flags
Accessors: Get
```

Return the flags of the annotation as 32 bit integer.

1 Invisible
2 Hidden (PDF 1.2)
3 Print (PDF 1.2)
4 NoZoom (PDF 1.3)
5 NoRotate (PDF 1.3)
6 NoView (PDF 1.3)
7 ReadOnly (PDF 1.3)
8 Locked (PDF 1.4)
9 ToggleNoView (PDF 1.5)

IsMarkup

```
Property Booloean IsMarkup
Accessors: Get
```

Return whether the annotation is a markup annotation. The following annotations are considered markup annotations:

- Free Text annotations
- Annotations that have a pop-up window that may display text
- Sound annotations

Name

```
Property String Name
Accessors: Get
```

Return the name of the annotation as string.

Rect

```
Property Variant Rect
Accessors: Get
```

Return the rectangle of the annotation as x1, y1, x2, y2. Where x1, y1 is the lower left corner of the annotation and x2, y2 the upper right corner. The coordinates are raw pdf coordinates. In order to calculate where the rectangle is positioned on the page as displayed by a viewer, the rectangle must be cropped using the page's CropBox and rotated using the Rotate attribute.

Subj

```
Property String Subj
Accessors: Get
```

Return the text representing a short description of the subject. This property is only available for mark-up annotations (requires PDF 1.5 or later).

Subtype

```
Property String Subtype
Accessors: Get
```

Return the type of the annotation as string, such as "Widget", "Square", "PopUp", "FreeText", "Ink", etc.

TextLabel

```
Property String TextLabel
Accessors: Get
```

Return the text label of the annotation as string. This label is usually used for the name of the author.

URI

```
Property String URI
Accessors: Get
```

Return the URI entry of the annotation as string if present.

Vertices

```
Property Variant Vertices
Accessors: Get
```

Return the vertices of a polygon annotation.

4.12 OutlineItem Interface

Count

```
Property Long Count
Accessors: Get
```

Return the number of children of the current outline. A negative number means the child tree is not opened.

Dest

```
Property IPDFDestination* Dest
Accessors: Get
```

Return an interface to the destination (see Destination Interface).

Title

```
Property String Title
Accessors: Get
```

Return the title of the outline.

4.13 Destination Interface

Note that the properties Bottom, Left, Right and Top of the destination interface have different meanings depending on the Type of the destination. The coordinates are raw PDF user space coordinates.

Bottom

```
Property Single Bottom
Accessors: Get
```

Return the Bottom value.

Left

```
Property Single Left
Accessors: Get
```

Return the Left value.

PageNo

```
Property Long PageNo
```

```
Accessors: Get
```

Return the target page number.

Right

```
Property Single Right
Accessors: Get
```

Return the Right value.

Top

```
Property Single Top
Accessors: Get
```

Return the Top value.

Type

```
Property Single Type
Accessors: Get
```

Return the type of the destination, such as "XYZ", "Fit", "FitH", "FitR", etc.

Zoom

```
Property Single Zoom
Accessors: Get
```

Return the Zoom value of the destination. A value of 0 has means the zoom level is left as is. It has the same meaning as a null value, the returns value will be 0 in both cases. A value of 1 means 100% magnification.

4.14 Ocg Interface

The optional content group (OCG) interface allows to list optional content groups (also known as "Layers") and their properties.

Optional content groups (OCGs) in PDF differ substantially from the simple layer paradigm found e. g. in graphics editing programs. Graphics objects in PDF do not belong to an OCG. Instead, their visibility is calculated by a Boolean function dependent on the state of any number of OCGs. For example, a path could be visible only if OCG "A" is ON and OCG "B" is OFF.

The functionality of OCG are described in depth in ISO 32000-1, chapter 8.11.4 or in the PDF Reference, chapter 4.10. OCG is supported in PDF 1.5 or later.

In order to extract content from all layers, the IgnoreOCM property can be to true.

For more background information including a sample see the section Optional Content (Layers).

Label

```
Property Boolean Label

Accessors: Get
```

Flag that indicates, whether this is an OCG or a label. Labels are used to label groups of OCGs in the hierarchy. Setting their visibility has ho effect.

Level

```
Property Long Level

Accessors: Get
```

In user interfaces OCGs can be shown in a tree. The property level indicates the hierarchy level of the OCG in that tree. OCG with Level 0 is a top level OCG. Level -1 means, that the OCG is not part of the hierarchy, it should not be presented to the user. Parent elements in the OCG hierarchy can be labels or OCGs. If the level of a label b is higher than its predecessor a, b is the parent element of the following objects of the same level as b. If the level of an OCG b is higher than its predecessor ocg a, a is the parent of the following objects of the same level as b. Note that the hierarchy reflects actual nesting of OCGs in the content. Setting the visibility of an OCG to true only has an effect, if the visibilities of all its parents are set to true.

Name

```
Property String Name

Accessors: Get
```

Return the name of the OCG.

Visible

```
Property Boolean Visible

Accessors: Get, Set
```

Get or set if the OCG is visible. This property controls the extraction of content objects. The default value is the one configured in the PDF document.

Note that though invisible paths generate no marks on the page, they still have an effect on the graphics state. For example their effect on the current drawing position and the clipping region does not change. Therefore, all paths are "active" and extracted regardless of their visibility. Invisible paths just use the end path operator "n", instead of a filling or stroking operator.

Example 1

id, OCGs, Level:	Hierarchy
0, OCG A, 0	- OCG A
1, OCG B, 0	- OCG B
2, OCG B1, 1	-- OCG B1
3, OCG B2, 1	-- OCG B2
4, OCG C, -1	hidden: OCG C

Example 2

id, OCGs/Labels, Level	Hierarchy
0, OCG A, 0	- OCG A
1, Label B, 1	- Label B
2, OCG B1, 1	-- OCG B1
3, OCG B2, 1	-- OCG B2
4, Label C, 1	- Label C
5, OCG C1, 1	-- OCG C1
6, OCG D, 0	- OCG D

4.15 PDFObject Interface

This interface represents a basic PDF object. More information on these types of objects can be found in chapter 3.2 of the PDF Reference. The PDFObject interface represents an object, which can be one of eight types. Depending on its type, different methods and properties should be used.

Note: If PDF objects are traversed recursively, it must be ensured the program does not end up in an endless-loop for cyclical structures.

There is a Java sample 'PdfObjExt.java' available that shows how to use this interface.

Begin, GetNext, End

applies to Dictionaries

```
Property Long Begin
Property Long End
Method Long GetNext(Long i)
```

Iterator: Property *Begin*, method *GetNext*, and property *End* can be used to traverse a dictionary object. *GetKey* and *GetValue* return the key and value of an element.

C# Example:

```
for (int i = dict.Begin; i != dict.End; i = dict.GetNext(i))
```

```
{ /* do something */ }
```

BooleanValue

```
Property Boolean BooleanValue
Accessors: Get
```

Return the Boolean value of a Boolean object

Dispose, DestroyObject

.NET API:

All objects retrieved from the API are destroyed when the document is closed. However, it is recommended to use *Dispose* as soon as possible in order to save memory.

Java and C/C++ API:

The TPdfExpaPDFObject objects must always be deleted using ExpaPDFObjectDestroyObject.

GetElement

applies to Arrays

```
Method PDFObject* GetElement(Long i)
```

Return the element at the index.

GetEntry

applies to Dictionaries

```
Method PDFObject* GetEntry (String Name)
```

Return the entry of the dictionary.

GetStream

applies to Indirect Objects

```
Method PDFObject* GetStream (String FileName)
property Variant StreamMem()
```

Return the indirect object's stream, if present. If the object is an image, the compressed stream is returned, otherwise the stream is decompressed.

IntegerValue

```
Property Long IntegerValue
Accessors: Get
```

Return the integer value of a numeric object.

Name

```
Property String Name
Accessors: Get
```

Returns the character sequence of a name object. The string is null terminated.

ObjectNumber

applies to Indirect Objects

```
Method Long ObjectNumber
```

Return the object number.

RealValue

```
Property Double RealValue
Accessors: Get
```

Return the real value of a numeric object

Size

applies to Arrays

```
Property Long Size
Accessors: Get
```

Returns the size of the array.

StringValue

```
Property Variant StringValue
Accessors: Get
```

Return the content of a string object as byte array.

Type

```
Property Type Type
Accessors: Get
```

Return the type of the object. Possible return values: eTypeBoolean, eTypeInteger, eTypeReal, eTypeString, eTypeName, eTypeArray, eTypeDictionary, eTypeIndirect

4.16 EmbeddedFile Interface

CheckSum

```
Property Variant CheckSum
```

```
Accessors: Get
```

Get the 16-byte MD5 check sum.

CreationDate

```
Property String CreationDate
Accessors: Get
```

Get the creation date.

FileName

```
Property String FileName
Accessors: Get
```

Get the embedded file's path. If the embedded file has no associated file stream (the functions *Store()* and *StoreInMemory()* return false), the *FileName* property references an external file.

ModDate

```
Property String ModDate
Accessors: Get
```

Get the modification date.

Store

```
Method Boolean Store(String Path)
```

Store the embedded file to disk.

- Parameters:

  Path: The file name and path, where the document shall be stored

- Return Values:

  True: if the operation competed successfully,

  False otherwise

StoreInMemory

```
Method Variant StoreInMemory()
```

Store the embedded file in memory.

- Return Values:

  The embedded file as a byte array.

4.17 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).

TPDFCompression

eComprRaw	No compression
eComprJPEG	Joint Photographic Expert Group
eComprFlate	Flate compression
eComprLZW	Lempel-Ziv-Welch
eComprGroup3	CCITT Fax Group 3
eComprGroup3_2D	CCITT Fax Group 3 2D
eComprGroup4	CCITT Fax Group 4
eComprJBIG2	Joint Bi-level Image Experts Group
eComprJPEG2000	JPEG2000
eComprUnknown	Unknown compression
eComprDefault	Apply a default compression which suites the color space of the image

Note that not all image formats/color depths support all compression types.

TPDFContentObject

See also function Content.GetNextObject.

eBeginOCM	Start of a sequence of objects, whose visibility is defined by an optional content membership string.
eEndOCM	End of OCM sequence
eNone	No content object
eText	Text object
eImage	Image object
ePath	Path object
eSave	Save the current graphics state
eRestore	Restore the current graphics state

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: **S**uccess, **E**rror, **W**arning, **I**nformation. 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*. The error codes that are listed to file access are listed here.

PDF_S_SUCCESS	The operation was completed successfully.
PDF_E_EVAL	This software is an evaluation version. Please contact www.pdf-tools.com.
PDF_E_FILEOPEN	The file couldn't be opened.
PDF_E_FILECREATE	The file couldn't be created.
PDF_E_PASSWORD	The authentication failed due to a wrong password.

TPDFOrientation

eOrientationUndef	Undefined
eOrientationTopLeft	Pages appear in columns, from bottom to top and right to left relative to page orientation.
eOrientationTopRight	Pages appear in columns, from bottom to top and left to right relative to page orientation.
eOrientationBottomRight	Pages appear in columns, from top to bottom and left to right relative to page orientation.
eOrientationBottomLeft	Pages appear in columns, from top to bottom and right to left relative to page orientation.
eOrientationLeftTop	Pages appear in rows, from right to left and bottom to top relative to page orientation.
eOrientationRightTop	Pages appear in rows, from left to right and bottom to top relative to page orientation.
eOrientationRightBottom	Pages appear in rows, from left to right and top to bottom relative to page orientation.
eOrientationLeftBottom	Pages appear in rows, from right to left and top to bottom relative to page orientation.

TPDFTextExtractConfiguration

eTECBreakTextState	Start new text object, if text state changes (font, font size, horizontal scaling). Set this property, if text state is important to you.
eTECBreakGraphicsState	Start new text object, if graphics state changes (color). Set this option, if the color is important to you.
eTECBreakSpaceUnicode	Start new text object, if extracted text contains a blank Unicode (\t, ` `, nbsp, etc.). Do not set this option, if you need the RawString property. Example: If set, the text "Hello World" will be extracted as "Hello" and "World" and otherwise as "Hello World"
eTECPosMergeSingleSpace	Merge text tokens that are a single space width apart (displacement), insert space. Do not set this option, if you need the RawString property.

	Example: If set, the text objects "Hello" and "World" are extracted as "Hello World", if they are approximately one space width apart.
eTECPosMergeMultiSpace	Merge text tokens that are one or more space widths apart (displacement), insert multiple spaces. Do not set this option, if you need the RawString property. Example: If set, the text objects "Hello" and "World" are extracted as "Hello World", where spaces are inserted to represent the distance of the objects.

5 Interface Changes

5.1 Changes from 1.4 to 1.4.1

This is a list of interface changes from version 1.4 (1.4.0.21) to version 1.4.1 (1.4.1.24).

Annotation Interface	New: Property *TextLabel*
ColorSpace Interface	New: Property *Colorant*
Content Interface	New: Property *Flags*
Destination Interface	New: Property *Zoom*
Font Interface	Removed: Property *FirstChar*, Property *LastChar*
Image Interface	New: Method StoreInMemory, Method GetImage
Page Interface	New: Property *BleedBox*, Property *TrimBox*, Property *ArtBox*, Property *DeviceColorant*
Text Interface	New: Property *BoundingBox*, Property *FontSize*, Property *Length*, Property *Rotation*, Property *Width*, Property *XPos*, Property *YPos* Removed: Property *RawString*, Property *TextMatrix*, Property *NextXPos*, Property *NextYPos* The properties *TexMatrix*, *NextXPos*, *NextYPos* are marked as deprecated.

No changes in the following interfaces: *AlternateImage*, *Document*, *GraphicsState*, *OutlineItem*, *TransformationMatrix*

5.2 Changes from 1.4.1 to 1.5

This is a list of interface changes from version 1.41 (1.4.1.24) to version 1.5 (1.5.0.40).

Annotation Interface	New: Property *Subj*, Property *Dest,* Property *URI*
ColorSpace Interface	New: Property *Colorant*
Content Interface	New: Property *SpaceFactor*
Document Interface	New: Method *GetDestination*, Property *IsLinearized*
Font Interface	Removed: Property *FirstChar*, Property *LastChar*
Text Interface	Removed: Property *TexMatrix*, Property *NextXPos*, Property *NextYPos*

5.3 Changes from 1.5 to 1.6

This is a list of interface changes from version 1.5 (1.5.0.40) to version 1.6 (1.6.0.41).

Annotation Interface	New: Property *Vertices*
ColorSpace Interface	New: Properties ColorantName, IsColor, IsMonochrome
Content Interface	New: Property BreakWords
Document Interface	New: Properties *Creator*, *Producer*
GraphicsState Interface	New: Properties *AlphaIsShape, BlendMode, FillAlphaConstant, FillOverprintFlag, FlatnessTolerance, OverprintMode, RenderingIntent, SmoothnessTolerance, StrokeAdjustment, StrokeAlphaConstant, StrokeOverprintFlag*
Font Interface	Changed: Type of *Flags* from *Long* to *int*
Image Interface	New: Properties IsBitonal, IsMonochrome, IsColor
Page Interface	New: Property *DeviceColorant*
Text Interface	New: Property *TextMatrix*

5.4 Changes from 1.6 to 1.7

This is a list of interface changes from version 1.6 (1.6.0.41) to version 1.7 (1.7.4.1).

Annotation Interface	New: Property *IsMarkup*
Document Interface	New: Method *GetPageLabel*

5.5 Changes from 1.7 to 1.8

This is list of interface changes from version 1.7 (1.7.4.1) to version 1.8 (1.8.35.1).

Image Interface	New: Property *SMask*

5.6 Changes from 1.8 to 1.9

This is list of interface changes from version 1.8 (1.8.35.1) to version 1.9 (1.9.24.1).

Document Interface	Deprecated: Property *ErrorCode*
	New: Property *LastError*
Content Interface	New: Property *ConvertPathToImage*
Colorspace Interface	Deprecated: Property *Colorant*
	New: Property *ColorantName*
	Deprecated: Property *High*
	New: Property *HighIndex*
Text Interface	Deprecated: Property *Length*
	New: Property *StringLength*

5.7 Changes from 1.9 to 1.91

This is list of interface changes from version 1.9 (1.9.24.1) to version 1.91 (1.91.28.0).

Content Interface	New: Properties *PathImageAntiAlias, PathImageBGColor, PathImageResolution, ConvertPathToImage*

5.8 Changes from 1.91 to 2.0

There are no interface changes from version 1.91 final to 2.0 final.

5.9 Changes from 2.0 to 2.1

The color profiles to transform RGB to CMYK values and vice versa when extracting colors in the directory bin\icc have been renamed from "CMYK.icc" and "sRGB.icm" to "USWebCoatedSWOP.icc" and "sRGB Color Space Profile.icm" to reflect their real names. The abbreviated version are no longer supported.

Document Interface	New: Methods *OcgCount, GetOcg*
	New: Property *LastErrorMessage, GetFirstEmbeddedFile, GetNextEmbeddedFile*
New: Interface Ocg	New: Properties *Label, Level*, *Name, Visible*
Content Interface	New: Properties *OCG*, *IgnoreOCG*
TPDFContentObject Enum	New: Enumerations eBeginOCM, eEndOCM
New: Interface PDFObject	New: Methods *GetElement, GetEntry, GetNext, GetStream*
	New: Properties *BooleanValue, IntegerValue, RealValue, StringValue, Name, Size, Begin, End, ObjectNumber, Type*
New: Interface EmbeddedFile	New: Methods *Store, StoreInMemory*
	New: Properties *CheckSum, CreationDate, FileName, ModDate*

5.10 Changes from 4.3 to 4.4

Content Interface	Removed: Properties *PathImageBGColor, PathImageAntiAlias, PathImageResolution, ConvertPathToImage*

5.11 Samples & Background Information

There are various code samples in the ZIP file of both, the evaluation and the release version of the 3-Heights™ PDF Extract Tool API.

Samples are also available at the website of PDF Tools for the 3-Heights™ PDF Extract Tool. Please find the latest samples online at: http://www.pdf-tools.com/asp/products.asp?name=EXPA

Note: Code samples in this manual are not constantly updated and might not be 100% compatible with the latest version of the Extract API.

5.12 Text Extraction

For text extraction a page number must be set. Using the method *GetNextText* returns the text tokens in Z order. This means the text token which is on top (i.e. is rendered last when the document is displayed) is retrieved last. Some PDF creators save the text in the order from the upper left to the lower right corner. As a result, extracting such documents, yields in a readable text sequence. This however is not true for all creators. It is as well possible to save every single character separately and in random order. Extracting text in such a document results in a random and therefore unreadable sequence of text tokens. The text tokens will first need to be sorted by coordinate in order too make it readable.

Undesired/Missing Blanks

Using the property *TextExtConfiguration* the text extraction algorithm can be configured. It is best to start with one of the settings recommended for your use case.

Sometimes this can lead to undesired blanks within what visually looks as one word. For example if:

- Text is written with different subsets of the same font. Different subsets of a font are considered different fonts. Therefore if the font changes within what visually looks as one word, it is separated.

- Text is not written on the same horizontal line. This can occur in some OCRed documents. There is a built-in tolerance to take account it this, however if Y-offsets are too large, a new word starts.

- Various possible errors in the font. Such as incorrect or missing width values of the glyphs (in particular of the blank), incorrect encoding, etc.

In all of the above cases, the coordinates need to be considered. Instead of inserting blanks after each word (as in the sample), the coordinate and width of the previous text token needs to be compared with the position of the next text token.

If text is concatenated, i.e. blanks are missing, decrease the property *SpaceFactor* for example to the value 0.2. (See also property *SpaceFactor* in the Content interface.)

Extracted Text is Unreadable

Fonts contain a particular set of glyphs. A glyph is a specific graphical rendering of a character. The glyphs P, **P** and *P* are glyphs of the character "P".

Fonts have an encoding, such as WinAnsi, or MacRoman, or custom encodings. The encoding maps the glyphs to a character. If the encoding in a font is missing, it is assumed it is WinAnsi encoded.

When an encoding is missing or incorrect, the text could become not extractable. Even if the text is visually readable, if the meaning of the glyphs is not encoded, it cannot be extracted (except by means of OCR).

If text is not extractable using the text extraction of Adobe Acrobat 7 Professional, then it's most likely not extractable with the 3-Heights™ PDF Extract Tool and vice versa.

Handling of Symbolic and Non-Symbolic Fonts

Fonts in PDF documents have so called font descriptor flags (See PDF Reference Manual, chapter 5.7.1). These flags describe the font characteristics, such as fixed pitch, serif, symbolic, italic, etc. If a font is flagged symbolic, it means its glyphs are not part of the standard Latin character set. Typical symbolic glyphs are squares, stars, or other small icons like cars or animals. Often there is no Unicode for these glyphs. The 3-Heights PDF Extract Tool handles text extraction of symbolic (as well as non-symbolic) fonts as described below.

If there is no encoding provided with the font, the intrinsic encoding is applied, which works as follows:

- In case font file is embedded:

  If there is a Unicode for the glyph, the corresponding Unicode is returned.

  If there is no Unicode and

  the font is flagged symbolic and part of the glyph names consist of a numerical value, such as G1, G2,… G100, the corresponding glyph number (and for TrueType fonts the Unicode Private Section prefix 0xF000) is returned. Otherwise the glyph index is returned.

  the font is non-symbolic, the standard encoding is used.

- In case font file is not embedded:

  The standard encoding is applied.

  Notes about the above algorithm:

  - When the standard encoding is applied, all control characters (<31) are mapped to character 32 (blank).

  - The glyph numbers G1, G2... G100 are often created by Ghost Script related PDF Creators. In these cases the number in the glyph name corresponds to the encoding of the used code page. E.g. G65 is the character A in WinAnsi encoding.

Text Extraction of Text Marked as Symbolic

Sometimes text is marked as symbolic, but it actually is not. In certain cases PDF creators do this to prevent text extraction. Assuming a PDF contains a TrueType font that is by mistake marked as symbolic. As a result the returned characters contain the Unicode Private Range prefix 0xF000 to 0xF0FF. In this case the prefix needs to be removed again. This can be achieved by setting the property *TranslateSymbolic* to true.

5.13 Image Extraction

Image extraction samples in different programming languages are available online at http://www.pdf-tools.com/pdf/pdf-extract-content-metadata-text.aspx.

An image is placed on the output page in any position, orientation, and size as specified by the current transformation matrix (property CTM of the current GraphicsState). The image space that is transformed by the CTM is the unit square [0 0 1 1], i.e. the unit square is mapped to the rectangle or parallelogram in which the image is to be painted. For example the coordinate on the page of the bottom right corner of the untransformed image is the transformation of the coordinate (1 1).

Image Resolution

Images are resources in a PDF document. Every image can be referenced multiple times in the document. The image itself doesn't have resolution, it only has a resolution when referenced on a page. The resolution depends on the ratio of the dimensions of the image and its size on the page, it can be different every time.

Image Orientation

Images can be stored with an orientation other than TopLeft (default). In order to display them visually correctly, there is a transformation matrix applied to invert the orientation. In order to ensure the images are saved with the same orientation as they are displayed on the PDF, use the method *ChangeOrientation* as shown in the sample.

5.14 Optional Content (Layers)

In order to associate content objects to Optional Content Groups (OCG) that define their visibility, the following steps have to be taken. First, the IgnoreOCM property must be set to true. Second, use the Content interface's GetNextObject() method to extract content objects. Whenever a BeginOCM operator is encountered, the OCM property contains the optional content membership string that defines the visibility of subsequent content objects, until the matching EndOCM operator is encountered. The respective OCG can be retrieved using the Document's GetOcg method.

As an example, look at file www.pdf-tools.com/public/downloads/samples/layers.pdf. It contains six colored squares and six optional content groups. The visibility of the red, green and blue squares is controlled by the respective OCGs. The yellow square is only visible, if both OCGs Green and Blue are ON. The OCGs "Gray 64" and "Gray 128" are child elements of the OCG "Gray" and control the visibility of the respective gray OCGs. These are visible only, if both the child and the parent OCG are ON.

Extracting OCGs from Layers.pdf:

id	name	level
0	*Red*	0
1	*Green*	0
2	*Blue*	0
3	*Gray*	0
4	*Gray 64*	1
5	*Gray 128*	1

Extracting objects from Layers.pdf:

type	property=value	comment
BeginOCM	*OCM="0"*	the visibility of subsequent objects is defined by the state of OCG 0 ("Red")
..Path	*Path=red square*	
EndOCM		end of OCM segment
BeginOCM	*OCM="1"*	OCG 1 is "Green"
..Path	*Path=green square*	
EndOCM		
BeginOCM	*OCM="2"*	OCG 2 is "Blue"
..Path	*Path=blue square*	
EndOCM		
BeginOCM	*OCM="1 && 2"*	subsequent objects are visible, if OCG 1 and OCG 2 are ON
..Path	*Path=yellow square*	
EndOCM		
BeginOCM	*OCM="3"*	OCG 3 is "Gray", parent OCG of 4 and 5
..BeginOCM	*OCM="4"*	note that OCM blocks can be nested, typically uses for hierarchical OCGs
....Path	*Path=gray 64 square*	
..EndOCM		
..BeginOCM	*OCM="5"*	OCG 5 is "Gray 128"
....Path	*Path=gray 128 square*	
..EndOCM		
EndOCM		