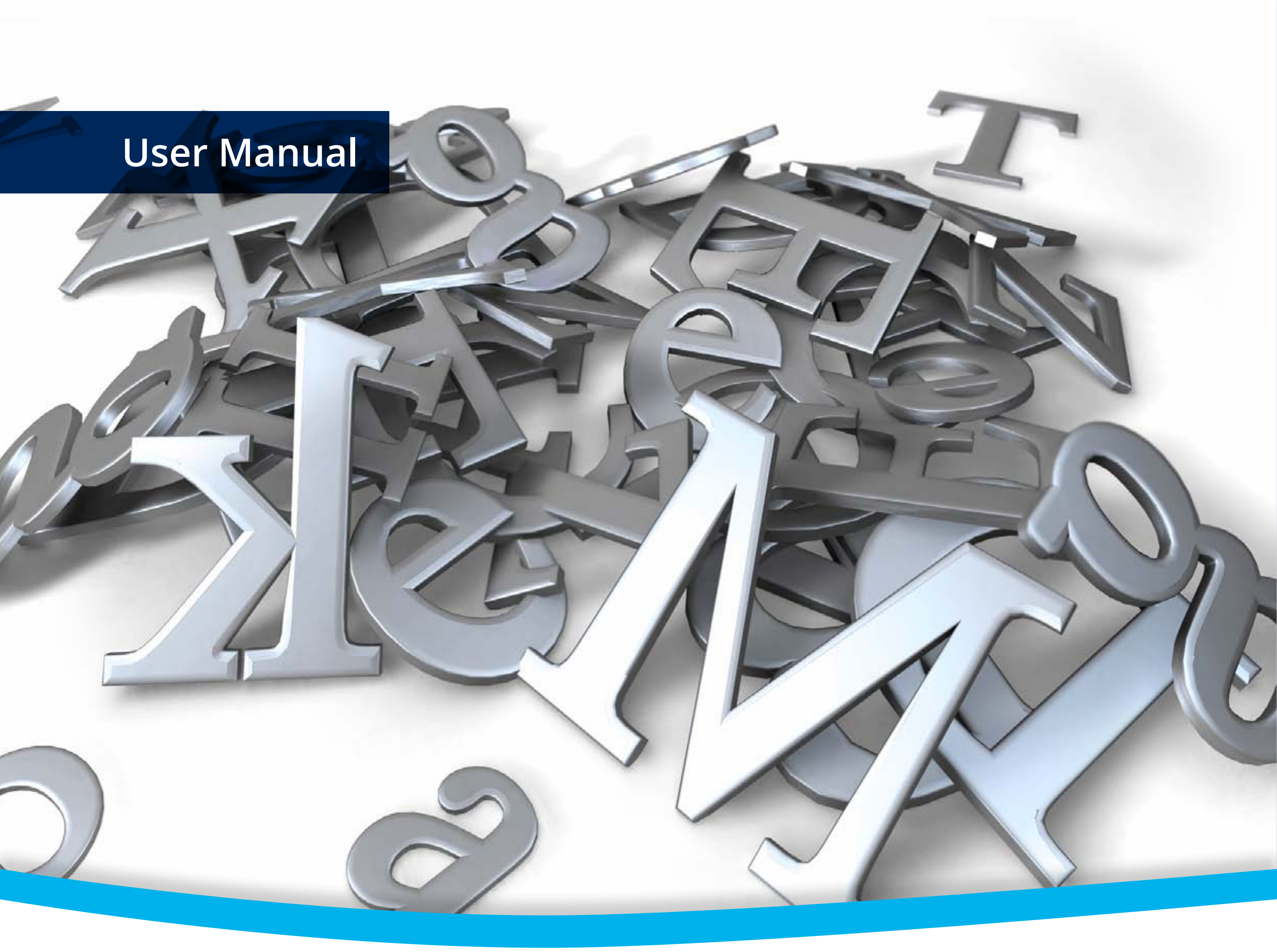
User Manual



3-Heights™ PDF Extract API

Version 4.10

PDF/A compliant

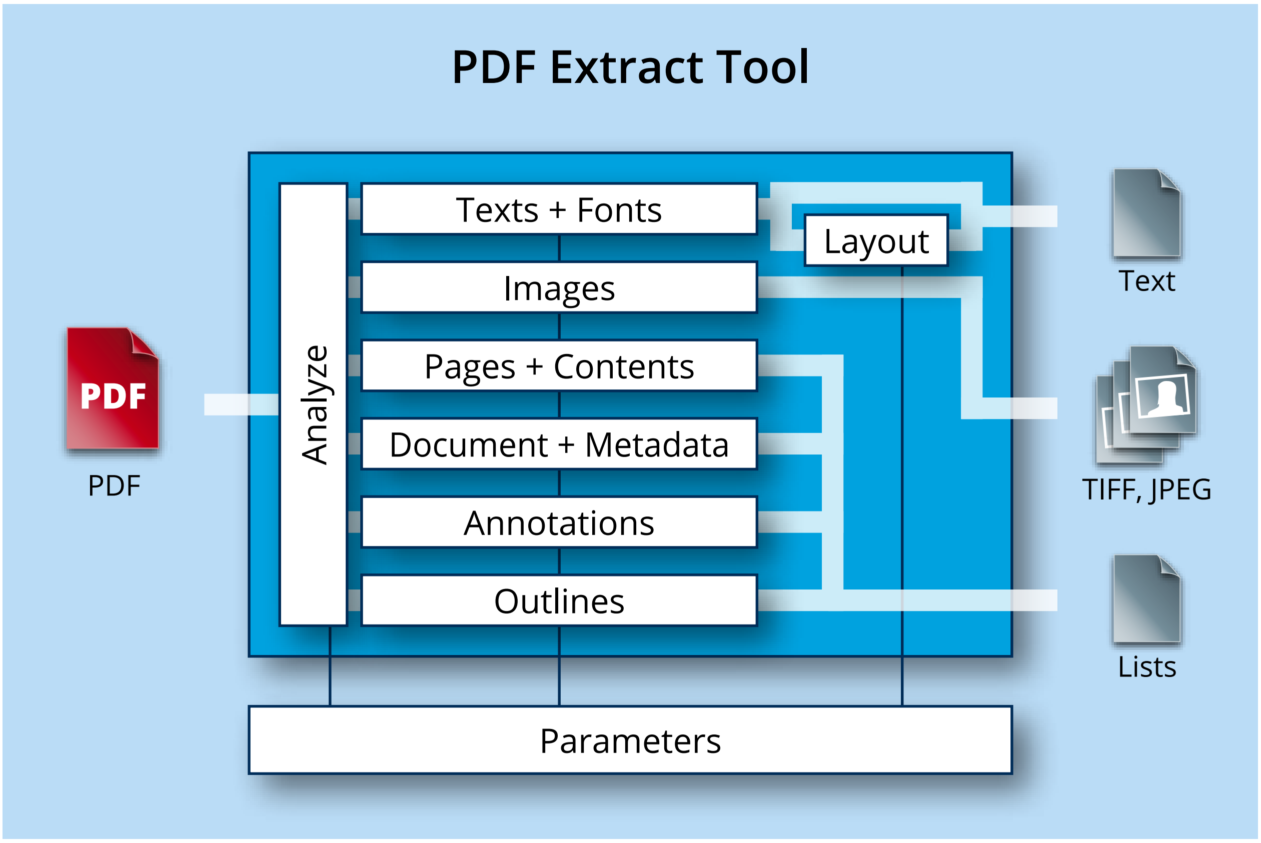
PDF-TOOLS.COM
Premium PDF Technology

Contents

1 Introduction

1.1 Description

The 3-Heights™ PDF Extract API is a tool for extracting and querying various attributes and page content from a PDF document. This includes texts, images, graphic objects, metadata, embedded fonts, and more, where some object types have additional properties to query. Configurable, intelligent mechanisms significantly increase extraction rates, for instance when extracting text.



1.2 Functions

The 3-Heights™ PDF Extract API 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.

1.2.1 Features

- Extract text:

    - Word by word with configurable word boundary detection
    - Retrieve text attributes such as position, font and font size
    - Automatically apply correct character decoding and produce Unicode output
    - Extract raw character codes
- Extract graphics objects (paths):
    - As strings that contain PDF graphics operators
    - Convert extracted paths to images
- Extract and store images:
    - Retrieve image attributes such as compression format, position and transparency masks
    - Extract and store transparency masks
    - Extract and store alternate images
- Extract PDF document-level information:
    - Page count
    - PDF version
    - Page labels
    - Creation and modification date
    - Document information such as title, author, subjects, and more
    - Outlines (bookmarks) including destinations
- Extract page information:
    - Media box, crop box, trim box, bleed box and art box
    - Page rotation
    - Annotations
- Extract and store embedded font files
- Retrieve detailed font information
- Retrieve optional content group (OCG) information and visibility (layers)
- Retrieve detailed graphic state information for each extracted page content object
- Extract raw PDF objects
- Retrieve detailed color space information including lookup tables for indexed color spaces
- Extract and store embedded files
- Specify a password to decrypt PDF files

1.2.2 Formats

Input Formats:

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

1.2.3 Compliance

- Standards: ISO 32000-1 (PDF 1.7)

1.3 Interfaces

The following interfaces are available:

- C
- Java
- .NET
- COM

1.4 Operating Systems

The 3-Heights™ PDF Extract API is available for the following operating systems:

- Windows 7, 8, 8.1, 10 – 32 and 64 bit
- Windows Server 2008, 2008 R2, 2012, 2012 R2, 2016 – 32 and 64 bit
- HP-UX 11i and later PA-RISC2.0 – 32 bit
- HP-UX 11i and later ia64 (Itanium) – 64 bit
- IBM AIX 6.1 and later – 64 bit
- Linux 2.6 – 32 and 64 bit
- Oracle 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)
- macOS 10.4 and later – 32 and 64 bit

2 Installation and Deployment

2.1 Windows

The 3-Heights™ PDF Extract API comes as a ZIP archive.

The installation of the software requires the following steps.

1. You need administrator rights to install this software.
2. Log in to your download account at `http://www.pdf-tools.com`. Select the product "PDF Extract API". If you have no active downloads available or cannot log in, please contact `pdfsales@pdf-tools.com` for assistance.
   You will find different versions of the product available. We suggest to download the version, which is selected by default. If another is required, it can be selected using the combo box.
   The product comes as a ZIP archive containing all files.
   There are 32 and 64-bit versions of the product available. While the 32-bit version runs on both, 32 and 64-bit platforms, the 64-bit version runs on 64-bit platforms only. The ZIP file contains both the 32-bit and the 64-bit version of the product.
3. Unzip the archive to a local folder, e.g. `C:\Program Files\PDF Tools AG\`.
   This creates the following subdirectories:

Subdirectory	Description
`bin`	Contains the runtime executable binaries.
`doc`	Contains documentation.
`include`	Contains header files to include in your C/C++ project.
`jar`	Contains Java archive files for Java components.
`lib`	Contains the object file library to include in your C/C++ project.
`samples`	Contains sample programs in various programming languages

4. (Optional) Register your license key using the License Management.
5. Identify which interface you are using. Perform the specific installation steps for that interface described in chapter Interface Specific Installation Steps
6. Make sure your platform meets the requirements regarding color spaces described in chapter Color Profiles.

2.2 Unix

This section describes installation steps required on all Unix platforms, which includes Linux, macOS, Oracle Solaris, IBM AIX, HP-UX, FreeBSD and others.

The Unix version of the 3-Heights™ PDF Extract API provides two interfaces:

- Java interface
- Native C interface

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

File Description

Name	Description
`bin/‹platform›/libPdfParser.so`	This is the shared library that contains the main functionality. The file's extension varies depending on the type of UNIX system. The directory `‹platform›` is either `x86` containing the 32-bit version of the library, or `x64` for the 64-bit version.
`doc/*.*`	Documentation
`include/*.h`	Contains header files to include in your C/C++ project.
`jar/EXPA.jar`	Java API archive.
`samples`	Example code.

2.2.1 All Unix Platforms

1. Unpack the archive in an installation directory, e.g. `/opt/pdf-tools.com/`
2. Copy or link the shared object into one of the standard library directories, e.g:

```
ln -s /opt/pdf-tools.com/bin/‹platform›/libPdfParser.so /usr/lib
```

3. Verify that the GNU shared libraries required by the product are available on your system:
   - *On Linux*:

   ```
   ldd libPdfParser.so
   ```

   - *On AIX*:

   ```
   dump -H libPdfParser.so
   ```

   In case the above reports any missing libraries you have two options:
   a. Use your system's package manager to install the missing libraries. On Linux it usually suffices to install the package `libstdc++6`.
   b. Use the PDF-Tools provided GNU shared libraries:
      1. Go to http://www.pdf-tools.com and navigate to "Support" →"Utilities".
      2. Download the GNU shared libraries for your platform.
      3. Extract the archive and copy or link the libraries into your library directory, e.g `/usr/lib` or `/usr/lib64`.
      4. Verify that the GNU shared libraries required by the product are available on your system now.
4. Optionally register your license key using the Command Line License Manager Tool.
5. Identify which interface you are using. Perform the specific installation steps for that interface described in chapter Interface Specific Installation Steps
6. Make sure your platform meets the requirements regarding color spaces described in chapter Color Profiles.

2.2.2 macOS

The shared library must have the extension `.jnilib` for use with Java. We suggest that you create a file link for this purpose by using the following command:

```
ln libPdfParser.dylib libPdfParser.jnilib
```

2.3 Interfaces

The 3-Heights™ 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 examples with which programming languages they can be used.

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 This interface is available in the Windows version only.
`Java`	The Java interface is available on all platforms.
`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 This interface is available in the Windows version only.
`C`	The native C interface is for use with C and C++. This interface is available on all platforms.

2.3.1 Development

The software developer kit (SDK) contains all files that are used for developing the software. The role of each file 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. See also table File Description to identify which files are required for your application.
Doc.	This file is for documentation only.
Empty field	An empty field indicates this file is not used at all for this particular interface.

Files for Development

Name	.NET	Java	COM	C
bin\‹platform›\PdfParser.dll	Req.	Req.	Req.	Req.
bin*NET.dll	Req.			
bin*NET.xml	Doc.			
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\‹platform›\PdfParser.lib				Req.
samples*.*	Doc.	Doc.	Doc.	Doc.

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

File Description

Name	Description
bin\‹platform›\PdfParser.dll	This is the DLL that contains the main functionality (required).
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 Visual Studio.
doc*.*	Various documentations.
include*.*	Contains files to include in your C / C++ project.
lib\‹platform›\PdfParser.lib	The object file library needs to be linked to the C/C++ project.
jar\EXPA.jar	The Java API archive.
samples*.*	Contains sample programs in different programming languages.

[1] These files must reside in the same directory as PdfParser.dll.

2.3.2 Deployment

For the deployment of the software only a subset of the files are required. Which files are required (Req.), optional (Opt.) or not used (empty field) for the four different interfaces is shown in the table below.

Files for Deployment

Name	.NET	Java	COM	C
`bin\‹platform›\PdfParser.dll`	Req.	Req.	Req.	Req.
`bin\*NET.dll`	Req.			
`jar\EXPA.jar`		Req.		

The deployment of an application works as described below:

1. Identify the required files from your developed application (this may also include color profiles).
2. Identify all files 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`. Color profiles are not used.
2. The application uses the COM interface and is distributed on Windows only.
   - The main DLL `PdfParser.dll` must be distributed.
3. All files are copied to the target location using a batch script. This script contains the following commands:

```
copy TextExt.exe %targetlocation%\.
copy PdfParser.dll %targetlocation%\.
```

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

```
regsvr32 /s %targetlocation%\PdfParser.dll.
```

2.4 Interface Specific Installation Steps

2.4.1 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. The following command shows the registration of PdfParser.dll. Note that in Windows Vista and later, the command needs to be executed from an administrator shell.

```
regsvr32 "C:\Program Files\PDF Tools AG\bin\‹platform›\PdfParser.dll"
```

Where `‹platform›` is `Win32` for the 32-bit and `x64` for the 64-bit version.

If you are using a 64-bit operating system and would like to register the 32-bit version of the 3-Heights™ PDF Extract API, you need to use the `regsvr32` from the directory `%SystemRoot%\SysWOW64` instead of `%SystemRoot%\System32`.[2]

If the registration process succeeds, a corresponding dialog window 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 reside in the same directory as the `PdfParser.dll`.

2.4.2 Java Interface

The 3-Heights™ PDF Extract API requires Java version 6 or higher.

For compilation and execution When using the Java interface, the Java wrapper `jar\EXPA.jar` needs to be on the CLASSPATH. This can be done by either adding it to the environment variable CLASSPATH, or by specifying it using the switch `-classpath`:

```
javac -classpath ".;C:\Program Files\PDF Tools AG\jar\EXPA.jar" sample.java
```

For execution Additionally the library `PdfParser.dll` needs be in one of the system's library directories[3] or added to the Java system property `java.library.path`. This can be achieved by either adding it dynamically at program startup before using the API, or by specifying it using the switch `-Djava.library.path` when starting the Java VM. Choose the correct subdirectory `x64` or `Win32` depending on the platform of the Java VM[4].

```
java -classpath ".;C:\Program Files\PDF Tools AG\EXPA.jar" ^
 -Djava.library.path=C:\Program Files\PDF Tools AG\bin\x64 sample
```

Note that on Unix-type systems, the path separator usually is a colon and hence the above changes to something like:

```
... -classpath ".:/path/to/EXPA.jar" ...
```

2.4.3 .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.

`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`.

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

[3] On Windows defined by the environment variable PATH and e.g. on Linux defined by LD_LIBRARY_PATH.

[4] If the wrong data model is used, there is an error message similar to this: `Can't load IA 32-bit .dll on a AMD 64-bit platform`

`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.

2.4.4 C Interface

- The header file `expa_c.h` needs to be included in the C/C++ program.
- The library `PdfParser.lib` needs to be linked to the project.
- The dynamic link library `PdfParser.dll` needs to be in a path of executables (e.g. on the environment variable `%PATH%`).

2.5 Uninstall, Install a New Version

If you have used the ZIP file for the installation: In order to uninstall the product, undo all the steps done during installation, e.g. un-register using `regsvr32.exe /u`, delete all files, etc.

Installing a new version does not require to previously uninstall the old version. The files of the old version can directly be overwritten with the new version.

2.6 Color Profiles

When extracting images, a color conversion might be neccessary.

For calibrated color spaces (such color spaces with an associated ICC color profile) the color conversion is well defined. For the conversion of uncalibrated device color spaces (DeviceGray, DeviceRGB, DeviceCMYK) however, the 3-Heights™ PDF Extract API requires apropriate color profiles. Therefore it is important, that the profiles are available and that they describe the colors of the device your input documents are intended for.

If no color profiles are available, default profiles for both RGB and CMYK are generated on the fly by the 3-Heights™ PDF Extract API.

2.6.1 Default Color Profiles

If no particular color profiles are set default profiles are used. For device RGB colors a color profile named `"sRGB Color Space Profile.icm"` and for device CMYK a profile named `"USWebCoatedSWOP.icc"` are searched for in the following directories:

Windows

1. `%SystemRoot%\spool\drivers\color`
2. directory `Icc`, which must be a direct sub-directory of where the `PdfParser.dll` resides.

Linux and other Unixes

1. `$PDF_ICC_PATH` if the environment variable is defined
2. the current working directory

2.6.2 Get Other Color Profiles

Most systems have pre-installed color profiles available, for example on Windows at `%SystemRoot%\system32\spool\drivers\color\`. Color profiles can also be downloaded from the links provided in the directory `bin\Icc\` or from the following websites:

- http://www.pdf-tools.com/public/downloads/resources/colorprofiles.zip
- http://www.color.org/srgbprofiles.html
- https://www.adobe.com/support/downloads/iccprofiles/iccprofiles_win.html

3 License Management

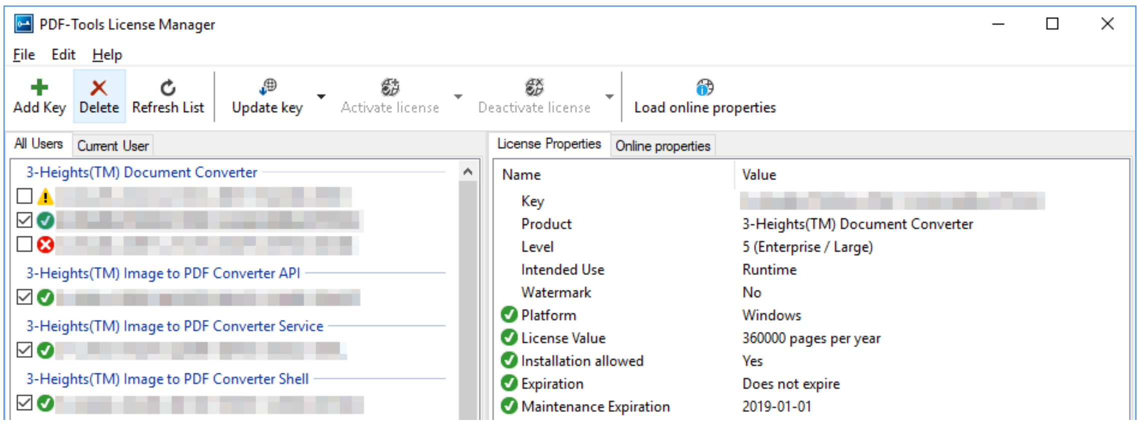
3.1 License Installation and Management

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 run-time via the `SetLicenseKey` method. This is the preferred solution for OEM scenarios.

3.1.1 Graphical License Manager Tool

The GUI tool `LicenseManager.exe` is located in the `bin` directory of the product kit (Windows only).


PDF-Tools License Manager
File
Edit
Help
Add Key
Delete
Refresh List
Update key
Activate license
Deactivate license
Load online properties
All Users
Current User
3-Heights(TM) Document Converter
3-Heights(TM) Image to PDF Converter API
3-Heights(TM) Image to PDF Converter Service
3-Heights(TM) Image to PDF Converter Shell
License Properties
Online properties

Name	Value
Key	[redacted]
Product	3-Heights(TM) Document Converter
Level	5 (Enterprise / Large)
Intended Use	Runtime
Watermark	No
✓ Platform	Windows
✓ License Value	360000 pages per year
✓ Installation allowed	Yes
✓ Expiration	Does not expire
✓ Maintenance Expiration	2019-01-01

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.

3.1.2 Command Line License Manager Tool

The command line license manager tool `licmgr` is available in the `bin\x86` and `bin\x64` directory.

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 is marked with a star '*' on the left side.

Add and delete license keys:

Install new license key:

```
licmgr store 1-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

Delete old license key:

```
licmgr delete 1-XXXXX-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

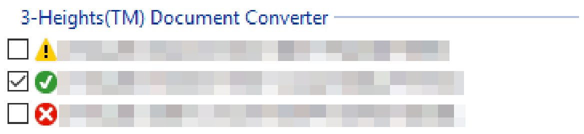
3.2 License Selection and Precedence

3.2.1 Selection

If multiple keys for the same product are installed in the same scope, only one of them can be active at the same time.

Installed keys that are not selected are not considered by the software!

In the Grahical User Interface use the check box on the left side of the license key to mark a license as selected.



With the Command Line Interface use the `select` subcommand:

```
licmgr select 1-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

3.2.2 Precedence

License keys are considered in the following order:

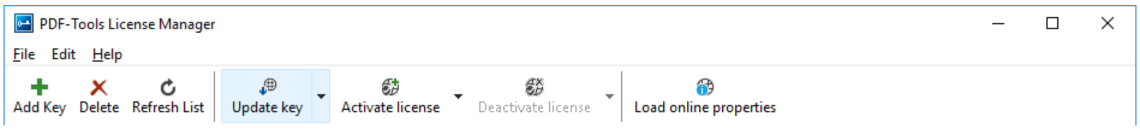
1. License key passed at runtime.
2. License selected for the current user
3. License selected for the current user (legacy key format)
4. License selected for all users
5. License selected for all users (legacy key format)

The first matching license is used, regardless whether it is valid or not.

3.3 Key Update

If a license property like the maintenance expiration date changes, the key can be update directly in the license manager.

In the Grahical User Interface select the license and press the button "Update Key" in the toolbar:



With the Command Line Interface use the `update` subcommand:

```
licmgr update 1-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

3.4 License activation

New licenses keys have to be activated (except for OEM licenses). These keys have to be installed in the license manager and may not be passed to the component at runtime.

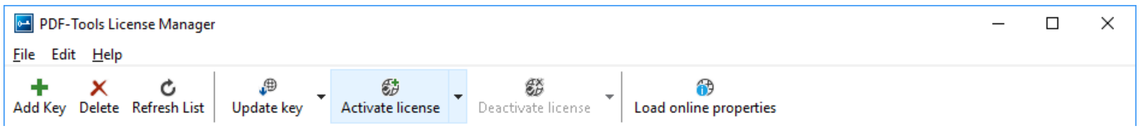
The license activation is tied to a specific computer. If the license is installed at user scope, the activation is also tied to that specific user. The same license key can be activated multiple times, if the license quantity is larger than 1.

Every license key includes a date, after which the license has to be activated, which is typically 10 days after the issuing date of the key. Prior to this date, the key can be used without activation and without any restrictions.

3.4.1 Activation

The License can be activated directly within the license manager. Every activation increases the activation count of the license by 1.

In the Grahical User Interface select the license and press the button "Activate license" in the toolbar:



With the Command Line Interface use the `activate` subcommand:

```
licmgr activate 1-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

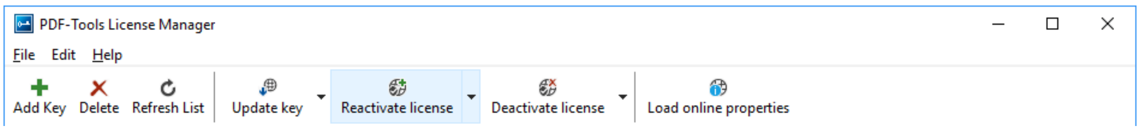
Note that the key has to be installed first.

3.4.2 Reactivation

The activation is tied to specific properties of the computer like the MAC address or host name. If one of these properties changes, the activation becomes invalid and the license has to be reactivated. A reactivation does **not** increase the activation count on the license.

The process for reactivation is the same as for the activation.

In the Grahical User Interface the button "Activate license" changes to "Reactivate license":



With the Command Line Interface the subcommand `reactivate` is used instead of `activate`:

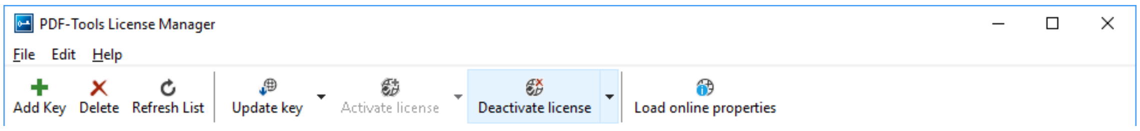
```
licmgr reactivate 1-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

3.4.3 Deactivation

To move a license to a different computer, it has to be deactivated first. Deactivation decreases the activation count of the license by 1.

The process for deactivation is similar to the activation process.

In the Grahical User Interface select the license and press the button "Deactivate license" in the toolbar:



With the Command Line Interface use the `deactivate` subcommand:

```
licmgr deactivate 1-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

3.5 Offline Usage

The following actions in the license manager need access to the internet:

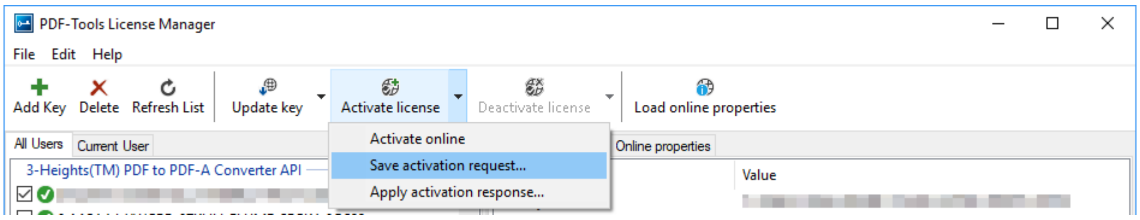
- License Activation
- License Reactivation

- License Deactivation
- Key Update

On systems wihout internet access, a three step process can be used instead, using a form on the PDF Tools website.

3.5.1 First Step: Create a Request File

In the Grahical User Interface select the license and use the dropdown menu on the right side of the button in the toolbar:



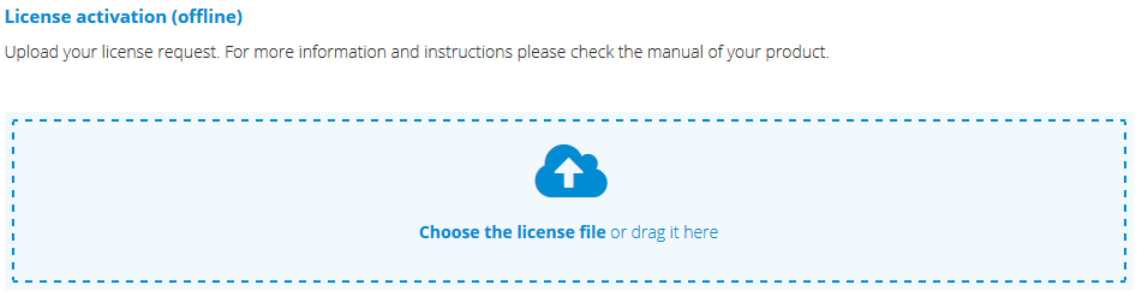
With the Command Line Interface use the `-fs` option to specify the destination path of the request file:

```
licmgr activate -fs activation_request.bin 1-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

License Deactivation: When saving the deactivation request file, the license is **deactivated immediately** and cannot be used any further. It can however only be activated again after completing the deactivation on the website.

3.5.2 Second Step: Use Form on Website

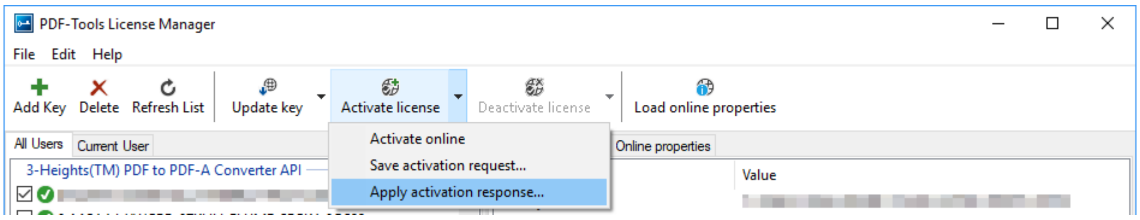
Open the following website in a web browser: `http://www.pdf-tools.com/pdf20/en/mypdftools/licenses-kits/license-activation/` Upload the request by dragging it onto the marked area:


License activation (offline)
Upload your license request. For more information and instructions please check the manual of your product.
Choose the license file or drag it here

Upon success, the response will be downloaded automatically if necessary.

3.5.3 Third Step: Apply the Response File

In the Grahical User Interface select the license and use the dropdown menu on right side of the button in the toolbar:



With the Command Line Interface use the `-fl` option to specify the source path of the response file:

```
licmgr activate -fl activation_response.bin 1-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

3.6 License Key Versions

As of 2018 all new keys will have the format `1-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX`. Legacy keys with the old format `0-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX` are still accepted for a limited time period.

For compatibility reasons, old and new version keys can be installed side by side and one key of each version can be selected at the same time. In that case, the software always uses the new version.

3.7 License Key Storage

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

3.7.1 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)

3.7.2 macOS

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)

3.7.3 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 solely 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
```

3.8 Troubleshooting

3.8.1 License key cannot be installed

The license key cannot be installed in the license manager application. The error message is: `"Invalid license format."`

Possible causes:

- The license manager application is an older version that only supports the legacy key format.

Solution

Use a current version of the license manager application or use a license key in the legacy key format if available.

3.8.2 License is not visible in license manager

The license key was successfully installed previously but is not visible in the license manager anymore. The software is still working correctly.

Possible causes:

- The license manager application is an older version that only supports the legacy key format.

Solution

Use a current version of the license manager application.

3.8.3 License is not found at runtime

The license is not found at runtime by the software. The error message is: `"No license key was set."`

Possible causes:

- The license key is actually missing (not installed).
- The license key is installed but not selected in the license manager.
- The application is an older version that only supports the legacy key format, while the license key has the new license format.

Solution

Install and select a valid license key that is compatible with the installed version of the software or use a newer version of the software. The new license key format is supported starting with version 4.10.26.1

For compatibility reasons, one license key of each format can be selected at the same time.

3.8.4 Eval watermark is displayed where it should not

The software prints an evaluation watermark onto the output document, even if the installed license is a productive one.

Possible causes:

- There is an evaluation license key selected for the **current user**, that takes precedence over the key for **all users**.

> **Note:** The software might be run under a different user than the license manager application.

- There is an evaluation license key selected with a newer license format that takes precedence over the key in the older format.
- The software was not restarted after changing the license key from an evaluation key to a productive one.

Solution

Disable or remove all evaluation license in all scopes and restart the software.

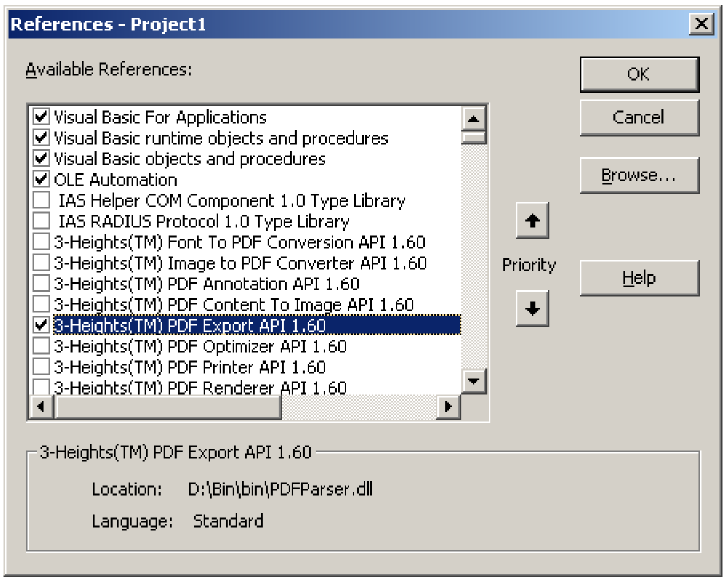
4 Programming Interfaces

4.1 Visual Basic 6

After installing the 3-Heights™ PDF Extract API and registering the COM interface (see Installation and Deployment), you find a Visual Basic 6 example with file extension `.vpb` in the directory `samples/VB/`. You can either use this sample as a base for an application, or you can start from scratch.

If you start from scratch, here is a quick start guide:

1. First create a new Standard-Exe Visual Basic 6 project. Then include the 3-Heights™ PDF Extract API component to your project.



References - Project1

Available References:

- [✓] Visual Basic For Applications
- [✓] Visual Basic runtime objects and procedures
- [✓] Visual Basic objects and procedures
- [✓] OLE Automation
- [ ] IAS Helper COM Component 1.0 Type Library
- [ ] IAS RADIUS Protocol 1.0 Type Library
- [ ] 3-Heights(TM) Font To PDF Conversion API 1.60
- [ ] 3-Heights(TM) Image to PDF Converter API 1.60
- [ ] 3-Heights(TM) PDF Annotation API 1.60
- [ ] 3-Heights(TM) PDF Content To Image API 1.60
- [✓] 3-Heights(TM) PDF Export API 1.60
- [ ] 3-Heights(TM) PDF Optimizer API 1.60
- [ ] 3-Heights(TM) PDF Printer API 1.60
- [ ] 3-Heights(TM) PDF Renderer API 1.60

3-Heights(TM) PDF Export API 1.60

Location: D:\Bin\bin\PDFParser.dll

Language: Standard

2. Draw a new Command Button and optionally rename it if you like.
3. Double-click the command button and insert the few lines of code below. All that you need to change is the path of the file name.

```
Private Sub Command1_Click()
  Dim doc As New PDFPARSERLib.Document

  If Not doc.Open("C:\path\input.pdf", "") Then
      MsgBox "Could not open PDF file. Error: " & doc.LastError
      Exit Sub
  End If

  MsgBox "Number of pages in the PDF: " & doc.PageCount

  doc.Close
End Sub
```

4.2 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.

Example:

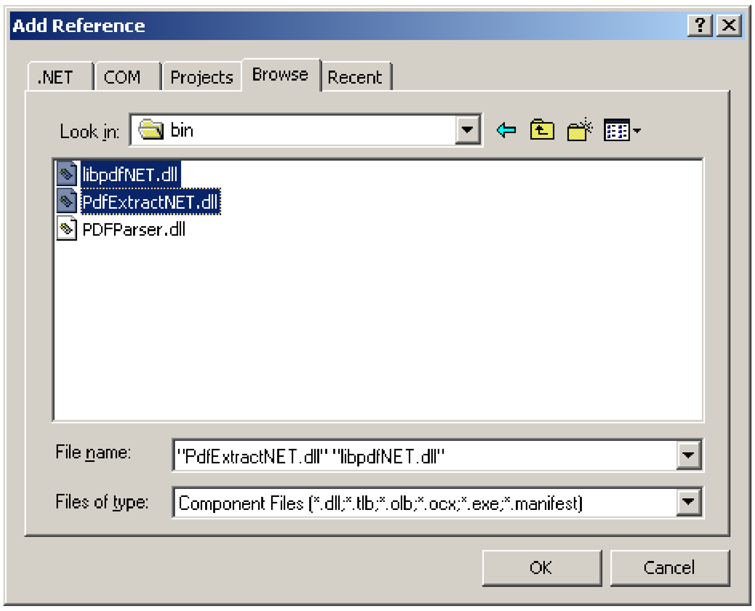
```
<%@ Language=VBScript %>
<%
     option explicit
     dim pdfDoc
     set pdfDoc = Server.CreateObject("PDFParser. Document")
     if not pdfDoc.Open(Server.Mappath("myfile.pdf")) then
       Response.Write "<p>"
       Response.Write "Could not open file." & "<br>"
     end if
     Response.Write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

4.3 .NET

There should be at least one .NET sample for MS Visual Studio available in the ZIP archive of the Windows version of the 3-Heights™ PDF Extract API. The 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 references 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`.
   Add them to the project as shown below:



libpdfNET.dll
PdfExtractNET.dll
PDFParser.dll

File name: "PdfExtractNET.dll" "libpdfNET.dll"

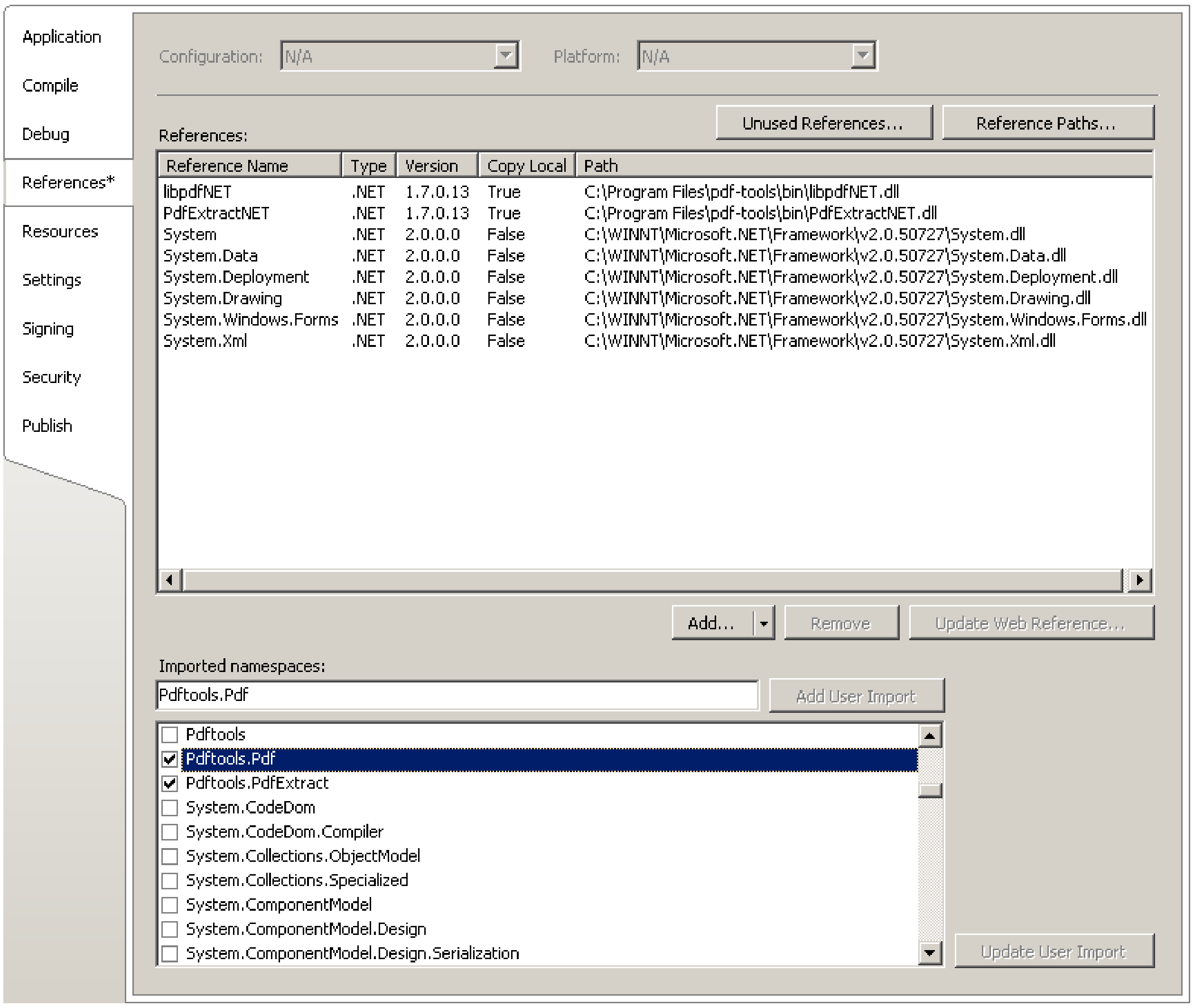
Files of type: Component Files (*.dll;*.tlb;*.olb;*.ocx;*.exe;*.manifest)

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

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

4.3.1 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`, `Pdftools.PdfRenderer`, and `Pdftools.PdfExtractNET`.
   You should now have settings similar as in the screenshot below:



Configuration: N/A Platform: N/A

References:

Reference Name	Type	Version	Copy Local	Path
libpdfNET	.NET	1.7.0.13	True	C:\Program Files\pdf-tools\bin\libpdfNET.dll
PdfExtractNET	.NET	1.7.0.13	True	C:\Program Files\pdf-tools\bin\PdfExtractNET.dll
System	.NET	2.0.0.0	False	C:\WINNT\Microsoft.NET\Framework\v2.0.50727\System.dll
System.Data	.NET	2.0.0.0	False	C:\WINNT\Microsoft.NET\Framework\v2.0.50727\System.Data.dll
System.Deployment	.NET	2.0.0.0	False	C:\WINNT\Microsoft.NET\Framework\v2.0.50727\System.Deployment.dll
System.Drawing	.NET	2.0.0.0	False	C:\WINNT\Microsoft.NET\Framework\v2.0.50727\System.Drawing.dll
System.Windows.Forms	.NET	2.0.0.0	False	C:\WINNT\Microsoft.NET\Framework\v2.0.50727\System.Windows.Forms.dll
System.Xml	.NET	2.0.0.0	False	C:\WINNT\Microsoft.NET\Framework\v2.0.50727\System.Xml.dll

Imported namespaces:

Pdftools.Pdf

- ☐ Pdftools
- ☑ Pdftools.Pdf
- ☑ Pdftools.PdfExtract
- ☐ System.CodeDom
- ☐ System.CodeDom.Compiler
- ☐ System.Collections.ObjectModel
- ☐ System.Collections.Specialized
- ☐ System.ComponentModel
- ☐ System.ComponentModel.Design
- ☐ System.ComponentModel.Design.Serialization

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

Example:

```
Dim document As New Pdftools.PdfExtract.Document()
document.Open(...)
document.PageNo = 1
Dim content = document.Page.Content
...
```

4.3.2 C#

3. Add the following namespaces:

Example:

```
using Pdftools.Pdf;
using Pdftools.PdfRenderer;
using Pdftools.PdfExtractNET;
```

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

Example:

```
using (Document document = new Document())
{
    document.Open(...);
    document.PageNo = 1;
    Content content = document.Page.Content;
    ...
}
```

4.3.3 Deployment

This is a guideline on how to distribute a .NET project that uses the 3-Heights™ PDF Extract API:

1. The project must be compiled using Microsoft Visual Studio. Hereby it is crucial that depending on the solution platform (x86 or x64) the matching native DLL `PdfParser.dll` (from the directory `bin\Win32` or `bin\x64`) is copied to the output directory.
2. The executable is created in the directory `bin\Release`.
3. For deployment, the executable and all .NET assemblies must be copied into the same folder on the target computer. The .NET assemblies of the 3-Heights™ PDF Extract API have the file name `bin\*NET.dll`.
4. At runtime, the native DLL `PdfParser.dll` must be found on the target computer by the DLL search sequence. To ensure this, the DLL must either be copied to the folder containing the executable or to a directory on the environment variable `Path` (e.g. `%SystemRoot%\system32`).
5. If required by the application, optional DLLs must be copied to the same folder. See Deployment for a list and description of optional DLLs.

4.3.4 Troubleshooting: TypeInitializationException

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

This exception can have two possible causes, distinguishable by the inner exception (property `InnerException`):

`System.DllNotFoundException` Unable to load DLL `PdfParser.dll`: The specified module could not be found.

`System.BadImageFormatException` An attempt was made to load a program with an incorrect format.

The following sections describe in more detail, how to resolve the respective issue.

Troubleshooting: DllNotFoundException

This means, that the native DLL `PdfParser.dll` could not be found at execution time.

Resolve this by either:

- adding `PdfParser.dll` as an existing item to your project and set its property "Copy to output directory" to "Copy if newer", or
- adding the directory where `PdfParser.dll` resides to the environment variable `%Path%`, or
- copying `PdfParser.dll` to the output directory of your project.

Troubleshooting: BadImageFormatException

The exception means, that the native DLL `PdfParser.dll` has the wrong "bitness" (i.e. platform 32 vs. 64 bit). There are two versions of `PdfParser.dll` available: one is 32-bit (directory `bin\Win32`) and the other 64-bit (directory `bin\x64`). It is crucial, that the platform of the native DLL matches the platform of the application's process.

The platform of the application's process is defined by the project's platform configuration for which there are 3 possibilities:

AnyCPU This means, that the application will run as a 32-bit process on 32-bit Windows and as 64-bit process on 64-bit Windows. When using AnyCPU one has to use a different native DLL, depending on the platform of Windows. This can be ensured either when installing the application (by installing the matching native DLL) or at application start-up (by determining the application's platform and ensuring the matching native DLL is loaded).

x86 This means, that the application will always run as 32-bit process, regardless of the platform of the Windows installation. The 32-bit DLL runs on all systems, which makes this the simplest configuration. Hence, if an application needs to be portable and does not require any specific 64-bit features, it is recommended to use this setting.

x64 This means, that the application will always run as 64-bit process. As a consequence the application will not run on a 32-bit Windows system.

5 User's Guide

There are various code samples in the product download kit of the 3-Heights™ PDF Extract API.

5.1 Text Extraction

For text extraction a page number must be set using the `PageNo` property. Then, 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 is however not true for all creators. It is as well possible to save every single character separately and in random order. Extracting text from 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 the text readable.

5.1.1 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 a configuration 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 documents whose text stems from OCR (optical character recognition). There is a built-in tolerance to take account of this. However, if vertical 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`.

5.1.2 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 can 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 text selection in the Adobe Reader, then it's most likely not extractable with the 3-Heights™ PDF Extract API and vice versa.

5.1.3 Handling of Symbolic and Non-Symbolic Fonts

Fonts in PDF documents have so called font descriptor flags (See PDF Reference 1.7, section 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 API 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 the 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 the 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 GhostScript-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.

5.1.4 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`.

Note: Since only the Unicode Private Range is affected by this translation, it is generally recommended to set this option to `True`.

5.2 Image Extraction

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 graphics state). 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).

5.2.1 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.

5.2.2 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.

5.3 Optional Content (Layers)

In order to associate content objects to Optional Content Groups (OCGs) 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 the document `www.pdf-tools.com/public/downloads/samples/layers.pdf`. It contains six colored squares and six OCGs. The visibility of the red, green and blue squares is controlled by the relevant 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"	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		

6 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.

6.1 Document Interface

6.1.1 Author

Property (get): `String Author`

Get the author entry from the document's info object. See `GetInfoEntry` for retrieving arbitrary entries from the info object.

6.1.2 Close

Method: `Void Close()`

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

6.1.3 Compliance

Property (get): `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.

6.1.4 CreationDate

Property (get): `Date CreationDate`

Get the creation date of the document's info object. See `GetInfoEntry` for retrieving arbitrary entries from the info object.

6.1.5 Creator

Property (get): `String Creator`

Get the name of the creator from the document's info object. See GetInfoEntry for retrieving arbitrary entries from the info object.

6.1.6 GetDestination

Method: `Destination GetDestination(String Destination)`

Retrieve an interface to the destination specified in the parameter.

Returns:

- A Destination Interface to the specified destination if it exists.
- Nothing otherwise.

Parameter:

Destination	[String]	The named destination

6.1.7 GetFirstColorSpaceResource, GetNextColorSpaceResource

Method: `ColorSpace GetFirstColorSpaceResource()`

Method: `ColorSpace GetNextColorSpaceResource()`

Use GetFirstColorSpaceResource to get the first color space resource. After this you can use GetNextColorSpaceResource to get further color space resources.

Returns:

- A ColorSpace Interface to the next color space resource if there is any.
- Nothing otherwise.

6.1.8 GetFirstEmbeddedFile, GetNextEmbeddedFile

Method: `EmbeddedFile GetFirstEmbeddedFile()`

Method: `EmbeddedFile GetNextEmbeddedFile()`

Use GetFirstEmbeddedFile to get the first embedded file. After this you can use GetNextEmbeddedFile to get further embedded files. Embedded files of both the document's collection (PDF Portfolio) and of FileAttachment annotations are returned.

Returns:

- An EmbeddedFile Interface to the next embedded file if there is any.

- Nothing otherwise.

6.1.9 GetFirstFontResource, GetNextFontResource

Method: `Font GetFirstFontResource()`

Method: `Font GetNextFontResource()`

Use GetFirstFontResource to get the first font resource. After this you can use GetNextFontResource to get further font resources.

Returns:

- A Font Interface to the next font resource if there is any.
- Nothing otherwise.

6.1.10 GetFirstImageResource, GetNextImageResource

Method: `Image GetFirstImageResource()`

Method: `Image GetNextImageResource()`

Use GetFirstImageResource to get the first image resource. After this you can use GetNextImageResource to get further image resources.

6.1.11 GetFirstOutlineItem, GetNextOutlineItem

Method: `OutlineItem GetFirstOutlineItem()`

Method: `OutlineItem GetNextOutlineItem(Long MaxLevel, Boolean ReturnOpenOnly)`

Method: `Long GetCurrentOutlineLevel()`

Use GetFirstOutlineItem to get the first outline item (bookmark). After this you can use GetNextOut-lineItem to get further outline items. After having called GetFirstOutlineItem or GetNextOutlineItem you can call GetCurrentOutlineLevel to get the level of the outline item in the hierarchy of outlines. A level of 0 corresponds to the root level.

Parameters:

MaxLevel	[Long]	(optional, default 20) The maximum level of the depth of the outlines.
ReturnOpenOnly	[Boolean]	(optional, default False) Return only outlines which are opened.

Returns:

- An OutlineItem Interface to the next outline item if there is any.

- `Nothing` otherwise.

6.1.12 `GetInfoEntry`

Method: `String GetInfoEntry(String szKey)`

Return the value of an entry in the document info dictionary. Popular entries specified in the PDF Reference 1.7 are `"Title"`, `"Author"`, `"Subject"`, `"Creator"` (sometimes referred to as Application), and `"Producer"` (sometimes referred to as PDF Creator). See PDF Reference 1.7 section "10.2.1 Document Information Dictionary" for more information about the document's info dictionary.

Parameter:

`szKey` `[String]` The string, such as `"Author"` or `"Subject"`, defining the entry in the document info dictionary.

Returns:

- The string corresponding to the entry if it exists.
- `Nothing` otherwise.

6.1.13 `GetPDFObject`

Method: `PDFObject GetPDFObject(String Path)`

Get a PDF object from a path string.

Parameter:

`Path` `[String]` The path consists of a prefix and operators. (In the following the symbols ‹ and › are used to mark a placeholder.)

Prefix:

- `"$/"`: Trailer dictionary (see section 3.4.4 of the PDF Reference 1.7), valid entries are `"$/Root"`, `"$/Info"`, and `"$/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"`

Returns:

A `PDFObject` Interface to the PDF object specified by the `Path` string.

6.1.14 `GetOcg`, `OcgCount`

Property (get): `Long OcgCount`

Method: `Ocg GetOcg(Long Count)`

Use `OcgCount` to get the number of optional content groups (OCGs), also known as "layers", of the document. An actual OCG can be retrieved with `GetOcg`.

Parameter:

`Count` `[Long]` The number of the optional content group. Optional content groups are numbered from 0 to OcgCount-1.

Returns:

A `Ocg` Interface to the requested OCG.

6.1.15 `GetPageLabel`

Method: `String GetPageLabel(Long PageNo)`

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

Parameter:

`PageNo` `[Long]` The page number

Returns:

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.

6.1.16 `GetXMPMetadata`

Method: `Boolean GetXMPMetadata(String FileName)`

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

Parameter:

`FileName` `[String]` The name of the output file

Returns:

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

6.1.17 GetXMPMetadataMem

Method: `Variant GetXMPMetadataMem()`

Extract the document's XMP metadata stream as a byte array. If the document does not contain XMP metadata, `Nothing` is returned.

6.1.18 IsCollection

Property (get): `Boolean IsCollection`

This property is `True` if the PDF document is a collection (aka PDF Portfolio) and `False` otherwise.

6.1.19 IsEncrypted

Property (get): `Boolean IsEncrypted`

This property is `True` if the PDF document has an encryption entry and `False` otherwise.

6.1.20 IsLinearized

Property (get): `Boolean IsLinearized`

This property is `True` if the linearization flag is set in the PDF document and `False` otherwise. 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.

6.1.21 Keywords

Property (get): `String Keywords`

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

6.1.22 `LastError`

Property (get): `TPDFErrorCode LastError`

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 `TPDFErrorCode`.

6.1.23 `LastErrorMessage`

Property (get): `String LastErrorMessage`

Get the error message text associated with the last error (see also `LastError`).

Note that the property is `Nothing` if no message is available.

6.1.24 `LicenseIsValid`

Property (get): `Boolean LicenseIsValid`

This property is `True` if a valid license key has been set or configured and `False` otherwise.

6.1.25 `MajorVersion, MinorVersion`

Property (get): `Short MajorVersion`

Property (get): `Short MinorVersion`

Get the major and minor version of the document. For example PDF Version 1.5 (which corresponds to Adobe Acrobat 6) yields `MajorVersion=1` and `MinorVersion=5`.

6.1.26 `ModDate`

Property (get): `Date ModDate`

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

6.1.27 `Open`

Method: `Boolean Open(String Filename, String Password)`

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

Parameters:

Filename [String] The file name and optionally the file path, drive or server string according to the operating systems file name specification rules.

Password [String] (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.

Returns:

True The file could be successfully opened.

False The file does not exist, it is corrupt, or the password is not valid. Use the property LastError for additional information.

6.1.28 OpenMem

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

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

Parameters:

MemBlock [Variant] The memory block containing the PDF file given as a one dimensional byte array.

Password [String] (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.

Returns:

True The document could be successfully opened.

False The document could not be opened, it is corrupt, or the password is not valid.

6.1.29 OpenStream

Method: `Boolean OpenStream(Variant Stream, String Password)`

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

Parameters:

Stream [Variant] The stream providing the PDF file. The stream must support random access.

`Password` `[String]` (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.

Returns:

`True` The document could be successfully opened.

`False` The document could not be opened, it is corrupt, or the password is not valid.

6.1.30 Page

Property (get): `Page Page`

This property allows to retrieve a Page Interface to the currently selected page of a document.

Note: Before querying this property, make sure that PageNo is set.

6.1.31 PageCount

Property (get): `Long PageCount`

Get the number of pages of an open document. If the document is closed or if the document is a collection (also known as PDF Portfolio) then this property is `0`.

6.1.32 PageNo

Property (get, set): `Long PageNo`

Set or get the currently selected page of an open document given its page number. If the document is closed this property evaluates to `0`.

The numbers are counted starting from `1` for the first page to the value of `PageCount` for the last page.

6.1.33 Producer

Property (get): `String Producer`

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

6.1.34 `ProductVersion`

Property (get): `String ProductVersion`

Get the version of the 3-Heights™ PDF Extract API in the format “A.C.D.E”.

6.1.35 `SetLicenseKey`

Method: `Boolean SetLicenseKey(String LicenseKey)`

Set the license key.

6.1.36 `Subject`

Property (get): `String Subject`

Get the subject from the document’s info object.

6.1.37 `Title`

Property (get): `String Title`

Get the title from the document’s info object.

6.2 Page Interface

6.2.1 `ArtBox`

Property (get): `Variant ArtBox`

Use this property to get the PDF “ArtBox” rectangle given by the coordinates left, bottom, right, top. The values are returned as an array of four single precision real numbers. The ArtBox is optional, it defines the region that contains meaningful content intended by the creator. If there is no ArtBox set, the CropBox is returned.

6.2.2 `BleedBox`

Property (get): `Variant BleedBox`

Use this property to get the PDF “BleedBox” rectangle given by the coordinates left, bottom, right, top. The values are returned as an array of four single precision real numbers. The BleedBox is optional, it defines the region to which the contents of the page should be clipped when output in a production environment. If there is no BleedBox set, the CropBox is returned.

6.2.3 `Content`

Property (get): `Content Content`

Access the `Content` Interface to the content stream of the page.

6.2.4 `CropBox`

Property (get): `Variant CropBox`

Use this property to get the PDF “CropBox” rectangle given by the coordinates left, bottom, right, top. The values are returned as an array of four single precision real numbers. The CropBox is optional, it defines the range of the visible region of the page. If there is no CropBox set, the MediaBox is returned.

6.2.5 `DeviceColorant`

Property (get): `String DeviceColorant`

Get the device colorant.

6.2.6 `Document`

Property (get): `Document Document`

Get the `Document` Interface to the document this page belongs to.

6.2.7 `GetFirstAnnotations, GetNextAnnotation`

Method: `Annotation GetFirstAnnotation()`

Method: `Annotation GetNextAnnotation()`

Use `GetFirstAnnotation` to get the first annotation. After this you can use `GetNextAnnotation` to get further annotations. Return an interface to the next annotation.

Returns:

- An Annotation Interface to the next annotation if any further annotations exist.
- `Nothing` otherwise.

6.2.8 HasColor

Property (get, set): `Boolean HasColor`

This property is `True` if the page contains color and `False` otherwise.

6.2.9 MediaBox

Property (get): `Variant MediaBox`

Use this property to get the PDF “MediaBox” 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.

Usually the MediaBox is rotated by viewer applications according to the property `Rotate`.

6.2.10 Rotate

Property (get): `Integer Rotate`

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`.

6.2.11 TrimBox

Property (get): `Variant TrimBox`

Use this property to get the PDF “TrimBox” rectangle given by the coordinates left, bottom, right, top. The values are returned as an array of four single precision real numbers. The TrimBox is optional, it defines the intended dimensions of the finished page after trimming. If there is no TrimBox set, the CropBox is returned.

6.3 Content Interface

The Content object provides information about the content of a PDF page. To obtain a Content object, use the `Content` method of the Page interface.

6.3.1 BreakWords

[Deprecated] **Property (get, set):** `Boolean BreakWords`

This property is deprecated. The former behavior of `BreakWords` can be simulated by setting the following flags in the `TextExtConfiguration` property:

- `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`.

6.3.2 BoundingBox

Property (get, set): `Variant BoundingBox`
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`.

6.3.3 ConvertPathToImage

Property (get, set): `Boolean ConvertPathToImage`
Default: `False`

This property works in conjunction with the method `GetNextObject`. When `ConvertPathToImage` is set to `True`, paths are converted and rendered and `GetNextObject` then returns images instead of paths.

`True`	Convert paths into images with an alpha channel.
`False`	Paths are returned natively.

Note: Images will contain a transparent background if the image format supports an alpha channel, such as TIF or PNG.

See also `PathImageAntiAlias`, `PathImageBGColor`, and `PathImageResolution`.

This property was first introduced in version 1.9.19.1. In version 4.6.26.0 it was re-implemented using the R2 rendering engine.

6.3.4 ExpandLigatures

Property (get, set): `Boolean ExpandLigatures`
Default: `False`

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

6.3.5 Flags

Property (get, set): Long Flags

If annotations are being extracted then this property accesses the annotation flags (see also property Flags in the Annotation Interface). Otherwise this property is -1.

6.3.6 GetNextImage

Method: Image 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 was found, an Image Interface to the object is returned. Alternatively, the same interface can be retrieved through the Image property. The corresponding graphics state can be retrieved through the GraphicsState property.

Returns:

- An Image Interface to the next image object on the current page if there is any.
- Nothing otherwise.

6.3.7 GetNextObject

Method: TPDFContentObject GetNextObject()

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

Returns:

eNone The end of the content stream has been reached.

eText A text object was found and its interface can be retrieved through the Text property.

eImage An image object was found and its interface can be retrieved through the Image property. The corresponding graphics state can be retrieved through the GraphicsState property.

ePath A path object was found and its string representation can be retrieved through the Path property. The corresponding graphics state can be retrieved through the 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 Marks the start of a sequence of objects, whose visibility is defined by optional content membership (OCM). The string representation of the OCM can be retrieved through the `OCM` property. OCM sequences can be nested.

eEndOCM Marks the end of an OCM sequence.

6.3.8 GetNextPath

Method: `String GetNextPath()`

This method reads the content stream objects until a path object is found or the end of the content stream is reached. If a path object was found, a string representation of a path object is returned. Alternatively, the same string can be retrieved through the `Path` property. The corresponding graphics state can be retrieved through the `GraphicsState` property.

Returns:

- A string representation of the next text path on this page if there is any.
- `Nothing` otherwise.

6.3.9 GetNextText

Method: `Text GetNextText()`

This method reads the content stream objects until a text object was found or the end of the content stream is reached. If a text object was found, a Text Interface to this object is returned. The corresponding graphic state can be retrieved through the `GraphicsState` property.

In contrast to the methods `GetNextImage` and `GetNextPath` this method may read several text objects and merge them until a major text property (font, line coordinate, etc.) changes or a word break occurs if word breaking is enabled. This merging behavior is configured with the properties `TextExtConfiguration` and `SpaceFactor`.

Returns:

- A Text Interface to the next text object if there is any on this page.
- `Nothing` otherwise.

6.3.10 GraphicsState

Property (get): `GraphicsState GraphicsState`

Get a GraphicsState Interface to the current graphics state.

The graphics state is updated if any of the following methods is called:

- GetNextText
- GetNextImage
- GetNextPath
- GetNextObject

6.3.11 IgnoreOCM

Property (get, set): `Boolean IgnoreOCM`

Set this property to `True` 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 set to `True`, hidden text is extracted as well. Set this property to `True` in order to extract all content.

6.3.12 Image

Property (get): `Image Image`

Get an Image Interface to the last read image object. The image object is updated each time the method `GetNextImage` is called or when `GetNextObject` returns `eImage`. The corresponding graphics state can be retrieved through the `GraphicsState` property.

6.3.13 OCM

Property (get): `String OCM`

Get the current optional content membership (OCM) string which defines the visibility as a Boolean function of OCG IDs in C syntax. Retrieve the respective OCG using the `GetOcg` method of the `Document` interface. Supported operators are `"&&"`, `"||"`, and `"!"`.

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 `GetNextObject` returned `eBeginOCM`.

6.3.14 Path

Property (get): `String Path`

Get a string representation of the last read path object. The path object is updated each time the method `GetNextPath` is called or when `GetNextObject` returns `ePath`. The corresponding graphics state can be retrieved through the `GraphicsState` property.

A 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 the PDF Reference 1.7.

Note: Calling the method `Reset` prior to extracting paths has an influence on the values of the coordinates of the path construction operators.

6.3.15 PathImageAntiAlias

Property (get, set): `Boolean PathImageAntiAlias`

Enables or disables anti-aliasing for the generated image when using `ConvertPathToImage`.

6.3.16 PathImageBGColor

Property (get, set): `Long PathImageBGColor`
Default: `0xFFFFFF`

Set or get the background color as RGB value of the generated image when using `ConvertPathToImage`. The format is `0x‹BB›‹GG›‹RR›`, where `‹BB›`, `‹GG›`, and `‹RR›` are hexadecimal numbers in the range `00-FF` that represent the values of the blue, green, and red channel respectively. The default `0xFFFFFF` is white.

Note: The image background is always transparent, or white for image formats such as JPEG that do not support transparency. The anti-aliasing will, however, be computed with the color specified in this property as a backdrop. Hence, this property only has an effect if `PathImageAntiAlias` is `True`.

6.3.17 PathImageResolution

Property (get, set): `Single PathImageResolution`
Default: `150`

Set or get the resolution in DPI of the generated image when using `ConvertPathToImage`.

6.3.18 Reset

Method: `Void Reset(Boolean AccountForRotate)`

This method resets the content extraction process, i.e. it sets the point of extraction to the beginning of the content stream.

Parameter:

`AccountForRotate` [`Boolean`] (Optional, default=`False`) This parameter allows to configure two modes for offsetting the coordinate system origin. This affects the coordinates of extracted content elements.

`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.

In both modes the unit of the coordinate system is 1/72 inch.

6.3.19 SpaceFactor

Property (get, set): `Long SpaceFactor`
Default: `0.3`

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.

6.3.20 Text

Property (get): `Text Text`

Get a `Text Interface` to the last read text object. The text object is updated each time the method `GetNextText` is called or when `GetNextObject` returns `eText`.

6.3.21 TextExtConfiguration

Property (get, set): `Long TextExtConfiguration`
Default: `eTECBreakTextState + eTECBreakGraphicsState + eTECBreakSpaceUnicode`

This property controls the way the text extraction algorithm works. When text is extracted, text objects are collected and merged 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`

6.3.22 TranslateSymbolic

Property (get, set): `Boolean TranslateSymbolic`
Default: `False`

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

Note: It is generally recommended to set this property to `True`.

6.4 Image Interface

6.4.1 Alternates

Property (get): `Variant Alternates`

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

Interface Note: Only in the COM interface does this property evaluate to an array of alternate images. In all other interfaces (C, .NET, and Java) this property returns a `AlternateArray` Interface through which alternate images can be accessed.

6.4.2 `BitsPerComponent`

Property (get): `Integer BitsPerComponent`

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.

6.4.3 `ChangeOrientation`

Method: `Boolean ChangeOrientation(TPDFOrientation iOrientation)`

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 `CTM`.

6.4.4 `ColorSpace`

Property (get): `ColorSpace ColorSpace`

Get a `ColorSpace` Interface to the color space of the image.

6.4.5 `Compression`

Property (get): `TPDFCompression Compression`

Get the compression used for the image in the PDF. See `TPDFCompression`.

6.4.6 `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 standard sRGB color space. Device color spaces are converted using pre-defined color profiles.

Returns:

`True`	If the conversion was successful.
`False`	Otherwise.

6.4.7 GetImage

Method: `Variant GetImage()`

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

Returns:

The image as a 1-dimensional `Byte` array.

6.4.8 GetResolution

Method: `Single GetResolution(TransformMatrix Matrix)`

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

Parameter:

`Matrix` [`TransformMatrix`] The transformation matrix of the image (see `TransformMatrix` Interface). 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.

Returns:

The calculated resolution in DPI.

6.4.9 Height

Property (get): `Long Height`

Get 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).

6.4.10 IsBitonal

Property (get): `Boolean IsBitonal`

This property is `True` when the image is bi-tonal.

6.4.11 IsColor

Property (get): `Boolean IsColor`

This property is True when the image is color.

6.4.12 IsMonochrome

Property (get): `Boolean IsMonochrome`

This property is True when the image is monochrome.

6.4.13 Mask

Property (get): `Image Mask`

Return the image's explicit mask if available. The mask image has one bit per pixel where a one bit indicates an opaque pixel and a zero bit indicates a transparent pixel. The resolution of the image and the image mask may differ. The image mask is placed in the same unit square in user space as the associated image.

6.4.14 ObjNumber

Property (get): `long ObjNumber`

Get 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.

6.4.15 Samples

Property (get): `Variant Samples`

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.

The interpretation of the sample data must be done according to the properties in the ColorSpace of the image.

6.4.16 SMask

Property (get, set): `Image SMask`

With this property the soft mask of an image can be extracted. If the image has a soft mask then this property provides an Image Interface to the soft mask. An image soft mask is a monochrome image. Sample values in the image do not represent grayscale pixels; rather, they designate the alpha value of the image. A value of 0 ("black") designates a "fully transparent" place and a value of 2^BitsPerComponent-1 ("white") "fully opaque". The resolution of the image and the image soft mask may differ. The image mask is placed in the same unit square in user space as the associated image.

6.4.17 `Store`

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

Store the image to a file.

Parameters:

`FileName` `[String]` The name of the disk file including drive, path, or server string according to the operating system's naming rules. The type of the image is defined by its extension:

Recognized Extensions

Extension	File Type
`".bmp"`, `".dib"`, `".pbm"`, `".pgm"`, `".pnm"`, `".ppm"`	Bitmap
`".jpg"`, `".jpe"`	JPEG
`".jb2"`	JBIG2
`".jpx"`, `".jp2"`	JPEG2000
`".tif"`	TIFF
`".png"`	PNG
`".gif"`, `".jif"`	GIF
`".eps"`	EPS

Use the second parameter `Compression` to indicate the compression type for TIFF files.

`Compression` `[TPDFCompression]` (optional) The compression type (for TIFF images). The default value is `eComprDefault`, which for TIFF images is LZW compression.

Returns:

`True` The file has successfully been written.

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

6.4.18 `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` `[String]` A file name extension that defines the type of image, see Recognized Extensions.

`Compression` `[TPDFCompression]` (optional) The compression type (for TIFF images). The default value is `eComprDefault`, which for TIFF images is LZW compression.

Returns:

`True` The image has successfully been saved.

`False` Otherwise.

6.4.19 `Width`

Property (get): `Long Width`

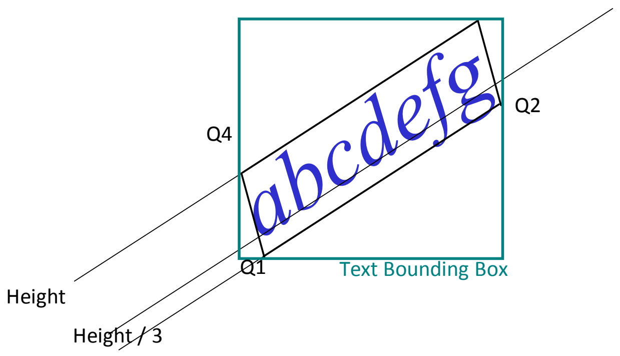
Return the width of the image in pixels (also called samples). The unit of pixels can be converted to a distance unit such as inch, millimeter etc. using a resolution value, i.e. 72 DPI (dots per inch).

6.5 Text Interface

6.5.1 `BoundingBox`

Property (get): `Variant BoundingBox`

Get 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.

6.5.2 `FontSize`

Property (get): `Single FontSize`

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

6.5.3 `Length`

[Deprecated] Property (get): `Short Length`

This property is superseded by `StringLength`.

6.5.4 `RawString`

Property (get): `Variant RawString`

For “simple fonts” this property returns the raw character codes from the PDF as a `Byte` array. For “CID fonts” (`Is-CID`) this property is `Nothing`. 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 a 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 that the creator has 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 following `TextExtConfiguration` flags, because the Unicode these flags work with might not be accurate:

- `eTECBreakSpaceUnicode`
- `eTECPosMergeSingleSpace`
- `eTECPosMergeMultiSpace`

6.5.5 `Rotation`

Property (get): `Single Rotation`

Get the rotation of the string in radians (rad). (2π rad = 360 °)

6.5.6 `StringLength`

Property (get): `Short StringLength`

Get the number of characters in the string.

6.5.7 UnicodeString

Property (get): `String UnicodeString`

Get 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 did not 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 did not embed Unicode mapping information for a symbolic font.

The `HasUnicodes` property of the `Font` Interface tells whether a text written with a specific font can be mapped to Unicodes.

6.5.8 TextMatrix

Property (get): `TransformMatrix TextMatrix`

Get a `TransformMatrix` Interface to the transformation matrix for this text.

> **Note:** This is not the TextMatrix in the PDF graphic state. This is a matrix that incorporates the current transformation matrix (CTM), the text matrix, and other text properties such as text rise and font size.

6.5.9 Width

Property (get): `Single Width`

Get the width of the string in points.

6.5.10 XPos, YPos, Position

Property (get): `Variant XPos`

Property (get): `Variant YPos`

Property (get): `TPDFVector Position`

> **Interface Note:** The properties `XPos` and `YPos` are only present in the COM interface. All other interfaces provide the property `Position`.

Get the horizontal and vertical position of the text's characters on the page in user coordinates. The return value is a 1-dimensional array holding the positions of all characters.

The length of the array is ‹n› + 1, where ‹n› is the length of the text, i.e., the value of `StringLength`. The zero-based index into the array has the following meaning:

`0` represents the first character.

`‹n›-1` represents the last character.

`‹n›` is a calculated, virtual position of where the next character would start. This position can be compared to the actual position of the next character in order to decide whether they belong to the same word or not.

`TPDFVector` represents a coordinate on the page and has two properties `x` and `y` of type `Float`.

6.6 `GraphicsState` Interface

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

The 3-Heights™ PDF Extract API 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 done using color profiles.

For in-detail information about, please refer to the PDF Reference 1.7 section "4.3 Graphic State".

6.6.1 `AlphaIsShape`

Property (get):	`Boolean AlphaIsShape`

Get the AlphaIsShape flag. It is `True` if the soft mask contains shape values, it returns false for opacity.

6.6.2 `BlendMode`

Property (get):	`String BlendMode`

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

6.6.3 `CharSpacing`

Property (get):	`Single CharSpacing`

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

6.6.4 CTM

Property (get): `TransformMatrix CTM`

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

6.6.5 DashArray

Property (get): `Variant DashArray`

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

6.6.6 DashPhase

Property (get): `Single DashPhase`

Get 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.

6.6.7 FillAlphaConstant

Property (get): `Single FillAlphaConstant`

Get the alpha constant for filling path painting operations. See the PDF Reference 1.7 Chapter 7 on transparency in general, and Section 4.5.3 for constant shape and constant opacity values which are linked to constant alpha values.

6.6.8 FillColorCMYK

Property (get): `Long FillColorCMYK`

Get the CMYK color quad that describes the color for filling operations. This value is obtained by converting the actual color by means of the `FillColorSpace`. The CMYK quads are encoded using the following formula:

Quad = $(((C \times 256) + M) \times 256 + Y) \times 256 + K$.

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

The CMYK values can be reconstructed as follows:

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.

6.6.9 FillColorRGB

Property (get): `Long FillColorRGB`

Get the RGB color triple that describes the color for filling operations. This value is obtained by converting the actual color 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.

6.6.10 FillColorSpace

Property (get): `ColorSpace FillColorSpace`

Get a ColorSpace Interface to the current color space that is used for filling operations. The color space is used to compute color values of the properties FillColorCMYK and FillColorRGB.

6.6.11 `FillOverprintFlag`

Property (get): `Boolean FillOverprintFlag`

Get the overprint flag for painting operations other than stroking. See the PDF Reference 1.7 Section 4.5.6 for more information about overprint.

6.6.12 `FlatnessTolerance`

Property (get): `Single FlatnessTolerance`

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

6.6.13 `Font`

Property (get): `Font Font`

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

6.6.14 `FontSize`

Property (get): `Single FontSize`

Get the current font size for text strings as a single precision real number in text units. It does not 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.

6.6.15 `HorizontalScaling`

Property (get): `Single HorizontalScaling`

Get 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.

6.6.16 `Leading`

Property (get): `Single Leading`

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

6.6.17 `LineCap`

Property (get): `Short LineCap`

Get 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

6.6.18 `LineJoin`

Property (get): `Short LineJoin`

Get 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

6.6.19 `LineWidth`

Property (get): `Single LineWidth`

Get the line width as a single precision real number in user units.

6.6.20 `MiterLimit`

Property (get): `Single MiterLimit`

Get 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.

6.6.21 `OverprintMode`

Property (get): `Short OverprintMode`

Get the overprint mode. See the PDF Reference 1.7 Section 4.5.6 for more information about overprint.

6.6.22 `RenderingIntent`

Property (get): `String RenderingIntent`

Get the name of the rendering intent.

6.6.23 `SmoothnessTolerance`

Property (get): `Single SmoothnessTolerance`

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

6.6.24 `SpaceWidth`

Property (get): `Float SpaceWidth`

Get the width of the space character in text space. To convert to page user units, transform this value using the text's transformation 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 `TextExtConfiguration` and `SpaceFactor`.

6.6.25 `StrokeAdjustment`

Property (get): `Boolean StrokeAdjustment`

Get the flag for the automatic stroke adjustment, a flag specifying whether to compensate for possible rasterization effects when stroking a path with a line width that is small relative to the pixel resolution of the output device. See PDF Reference 1.7 Section 6.5.4 for more information about stroke adjustment.

6.6.26 `StrokeAlphaConstant`

Property (get): `Single StrokeAlphaConstant`

Get the current alpha constant for stroking path painting operations. See the PDF Reference 1.7 Chapter 7 on transparency in general, and Section 4.5.3 for constant shape and constant opacity values which are linked to constant alpha values.

6.6.27 StrokeColorCMYK

Property (get): `Long StrokeColorCMYK`

Get the CMYK color quad that describes the color for stroking operations. This value is obtained by converting the actual color by means of the StrokeColorSpace. The CMYK quads are encoded using the following formula:

Quad = $(((C \times 256) + M) \times 256 + Y) \times 256 + K$.

6.6.28 StrokeColorRGB

Property (get): `Long StrokeColorRGB`

Get the RGB color triple that describes the color for stroking operations. This value is obtained by converting the actual color by means of the StrokeColorSpace. The RGB triples are encoded using the following formula:

Triple = $((R \times 256) + G) \times 256 + B$.

6.6.29 StrokeColorSpace

Property (get): `ColorSpace StrokeColorSpace`

Get a ColorSpace Interface to the current color space that is used for stroking operations. The color space is used to compute color values of the properties StrokeColorCMYK and StrokeColorRGB.

6.6.30 StrokeOverprintFlag

Property (get): `Boolean StrokeOverprintFlag`

Get the overprint flag for stroking painting operations. See the PDF Reference 1.7 Section 4.5.6 for more information about overprint.

6.6.31 TextKnockout

Property (get): `Boolean TextKnockout`

Get the text knockout flag. This Boolean flag determines the behavior of overlapping glyphs within a text object in the transparent imaging model.

6.6.32 TextRenderingMode

Property (get): `Short TextRenderingMode`

Get 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.

6.6.33 `TextRise`

Property (get): `Single TextRise`

Get 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.

6.6.34 `WordSpacing`

Property (get): `Single WordSpacing`

Get 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 1.7, section 4.3.)

6.7 Font Interface

6.7.1 `Ascent`

Property (get): `Single Ascent`

Get 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.

6.7.2 `AvgWidth`

Property (get): `Single AvgWidth`

Get the average width of the glyphs in the font.

6.7.3 BaseName

Property (get): `String BaseName`

Get the font name.

6.7.4 CapHeight

Property (get): `Single CapHeight`

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

6.7.5 Charset

Property (get): `String Charset`

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

6.7.6 Descent

Property (get): `Single Descent`

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

6.7.7 Encoding

Property (get): `Variant Encoding`

Method: `String Encoding(Long charIndex)`

Get the glyph name of characters.

Interface Note: The COM interface provides only the **property** `Encoding`. In this case the returned type is an array of length 256 of `String`s.
The other interfaces provide only the **method** `Encoding` and the return type is a `String`. The parameter `charIndex` specifies for which character to get the glyph name. This value must be between `0` and `255`.

6.7.8 Flags

Property (get): `Long Flags`

Get 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.

6.7.9 FontBBox

Property (get): `Variant FontBBox`

Get 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.

6.7.10 FontFile

Property (get): `Variant FontFile`

Get the font file as a `Byte` array. This property is `Nothing` if the font has no embedded font file.

6.7.11 FontFileType

Property (get): `Short FontFileType`

Get the type of the font file. See PDF Reference 1.7 section "5.7 Font Descriptors" for more information.

Value	PDF Font-File Type	Description
0	—	This font has no font file.
1	FontFile	Type 1 font.
2	FontFile2	TrueType font.
3	FontFile3	The type of the font is determined by Type.

6.7.12 HasUnicodes

Property (get): `Boolean HasUnicodes`

This property tells whether the font provides Unicodes. If this property is `False` then text written with this font is not extractable. I.e. the `UnicodeString` property of the `Text Interface` contains not Unicodes but directly the character IDs.

6.7.13 IsCID

Property (get): `Boolean IsCID`

This property is `True` if the font is a CID Font.

6.7.14 ItalicAngle

Property (get): `Single ItalicAngle`

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

6.7.15 Leading

Property (get): `Single Leading`

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

6.7.16 MaxWidth

Property (get): `Single MaxWidth`

Get the maximum width of the glyphs in the font.

6.7.17 `MissingWidth`

Property (get): `Single MissingWidth`

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

6.7.18 `StemH`, `StemV`

Property (get): `Single StemH`

Property (get): `Single StemV`

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

6.7.19 `Type`

Property (get): `Single Type`

Get the font type either of the following strings: `"Type1"`, `"Type1C"`, `"TrueType"`, `"Type3"`, `"Unknown"`.

6.7.20 `Widths`

Property (get): `Variant Widths`

Get an array of `Float` which contains the widths of the glyphs.

6.7.21 `WMode`

Property (get): `Long WMode`

Get the writing mode for this font. `0` means "horizontal writing mode" and `1` means "vertical writing mode".

The value of the property has both an effect on the font's glyph positioning as well as glyph appearance, e.g. parentheses are usually horizontal with WMode `0` and vertical with VMode `1`.

6.7.22 XHeight

Property (get): `Single XHeight`

Get 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 1.7, section 5.7.)

6.8 ColorSpace Interface

6.8.1 BaseColorSpace

Property (get): `ColorSpace BaseColorSpace`

Get a ColorSpace Interface to the base color space it exists. See also Lookup.

6.8.2 ColorantName

Property (get): `Variant ColorantName`

Get 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 an array containing all strings a single string is returned, e.g. `"Red"`.

6.8.3 ComponentsPerPixel

Property (get): `Integer ComponentsPerPixel`

Return the number of components per pixel.

6.8.4 HighIndex

Property (get): `Short HighIndex`

Get the highest possible index value for indexed colors. It is `0` when no indexed color space is used.

6.8.5 IsColor

Property (get): `Boolean IsColor`

This property is `True` when the color space has more than one channel.

6.8.6 IsIndexed

Property (get): `Boolean IsIndexed`

This property is `True` when the image uses indexed colors.

6.8.7 IsMonochrome

Property (get): `Boolean IsMonochrome`

This property is `True` when the color space is monochrome.

6.8.8 Lookup

Property (get): `Variant Lookup`

Get the color lookup table as a byte array. The lookup table is only present if the color spaced is an indexed (palleted) color space, i.e., if `IsIndexed` is `True`.

The lookup table contains color values in the `BaseColorSpace`. E.g., if the base color space is an RGB color space then the table contains bytes $R_0\ G_0\ B_0\ R_1\ G_1\ B_1 \ldots R_h\ G_h\ B_h$, where h is the value of `HighIndex`. The length of the table in bytes hence is $(h + 1) \cdot c$, where c is the value of `ComponentsPerPixel`. Refer to the PDF Reference 1.7 Section “4.5.5 Special Color Spaces” for more information about indexed color spaces.

6.8.9 Name

Property (get): `String Name`

Get the name of the color space as string, for example `"DeviceGrey"`, `"DeviceRGB"` or `"Indexed"`.

6.9 `TransformMatrix` Interface

6.9.1 a, b, c, d, e, f

Property (get):	`Single a`
Property (get):	`Single b`
Property (get):	`Single c`
Property (get):	`Single d`
Property (get):	`Single e`
Property (get):	`Single f`

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, the 3-Heights™ PDF Extract API also provides properties which compute these values.

The actual matrix is $M = \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ e & f & 1 \end{bmatrix}$

This matrix is used to define a transformation of a vector $[\,x\ y\ 0\,]$ to a vector $[\,x'\ y'\ 0\,] = [\,x\ y\ 0\,] \cdot M$, where (x, y) is the original point and (x', y') is the transformed point on the page. In the following we represent the matrix by the array [a b c d e f] holding the coefficients `a` to `f`.

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. The point (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 [cosα sinα -sinα cosα 0 0].

More detailed information can be found in the PDF Reference 1.7 sections "4.2.2 Common Transformations" and "4.2.3 Transformation Matrices".

6.9.2 Orientation

Property (get): `TPDFOrientation Orientation`

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

6.9.3 Rotation

Property (get): `Single Rotation`

Get the rotation angle of the matrix counter clockwise. This is equal to the minimum of `XSkew` and -`YSkew`.

6.9.4 XScaling, YScaling

Property (get): `Single XScaling`

Property (get): `Single YScaling`

Get the horizontal and vertical scaling factor.

6.9.5 XSkew, YSkew

Property (get): `Single XSkew`

Property (get): `Single YSkew`

Get the x and y-axis skewing. The transformation matrix [1 tanα tanβ 1 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.

6.9.6 XTranslation, YTranslation

Property (get): `Single XTranslation`

Property (get): `Single YTranslation`

Get the translation in horizontal and vertical direction. These are the same values as returned by the properties `e` and `f`.

6.10 AlternateImage Interface

6.10.1 DefaultForPrinting

Property (get): `Boolean DefaultForPrinting`

This property is `True` if the alternate image is set as default for printing.

6.10.2 Image

Property (get): `Image Image`

Get an `Image Interface` to the alternate image.

6.11 AlternateArray Interface

Interface Note: This interface exists only in the C, the .NET, and the Java interfaces. In the COM interface, the Alternates property returns directly an array of alternate images. In Java the name of this interface is AlternateImageArray.

6.11.1 Count

Property (get): `Integer Count`

Get the size of the array, i.e., the number of alternate images in the array.

6.11.2 GetElement

Method: `GetElement(Integer Index)`

Use this method to retrieve a AlternateImage Interface from the array.

Parameter:

Index	[Integer]	The zero-based index into the array.

6.12 Annotation Interface

For in-detail information about PDF annotations please refer to the PDF Reference 1.7 section "8.4 Annotations".

6.12.1 AttachedFile

Property (get): `EmbeddedFile AttachedFile`

Get an EmbeddedFile Interface to 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.

6.12.2 Color

Property (get): `Long Color`

Get to color of the annotation.

6.12.3 Contents

Property (get): `String Contents`

Get the content of the annotation.

6.12.4 Date

Property (get): `Date Date`

Get the date of the annotation. The used format is: #dd.mm.yyyy hh:mm:ss#

6.12.5 Dest

Property (get): `Destination Dest`

Get a `Destination` Interface to the destination of a link annotation. This entry is permitted if an A (action) entry is present.

6.12.6 Flags

Property (get): `Long Flags`

Get the flags of the annotation as 32 bit integer. The following bit-positions are possible:

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)

6.12.7 IsMarkup

Property (get): `Boolean IsMarkup`

Decides 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

6.12.8 Name

Property (get): `String Name`

Get the name of the annotation as string.

6.12.9 Rect

Property (get): `Variant Rect`

Get the rectangle of the annotation as an array [‹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.

6.12.10 Subj

Property (get): `String Subj`

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

6.12.11 Subtype

Property (get): `String Subtype`

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

6.12.12 `TextLabel`

Property (get): `String TextLabel`

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

6.12.13 `URI`

Property (get): `String URI`

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

6.12.14 `Vertices`

Property (get): `Variant Vertices`

Get the vertices of a polygon annotation.

6.13 `OutlineItem` Interface

6.13.1 `Count`

Property (get): `Long Count`

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

6.13.2 `Dest`

Property (get): `Destination Dest`

Get a `Destination` Interface to this outline's destination.

6.13.3 `Title`

Property (get): `String Title`

Get the title of the outline.

6.14 Destination Interface

For in-detail information about PDF destinations please refer to the PDF Reference 1.7 section “8.2.1 Destinations”.

> **Note:** 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.

6.14.1 Bottom

Property (get): `Single Bottom`

Get the Bottom value.

6.14.2 Left

Property (get): `Single Left`

Get the Left value.

6.14.3 PageNo

Property (get): `Long PageNo`

Get the target page number

6.14.4 Right

Property (get): `Single Right`

Get the Right value.

6.14.5 Top

Property (get): `Single Top`

Get the Top value.

6.14.6 Type

Property (get): `String Type`

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

6.14.7 Zoom

Property (get): `Single Zoom`

Get the Zoom value of the destination. A value of `0` means the zoom level is left unchanged. A value of `1` means 100% magnification.

6.15 Ocg Interface

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

The concept of OCGs in PDF differs substantially from the simple layer paradigm found e.g. in graphics editing programs. There is no one-to-one correspondence between graphics objects in PDF and OCGs. Instead, the object visibility is determined 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 OCGs are described in depth in ISO 32000-1, section 8.11.4 or in the PDF Reference 1.7, section 4.10. OCGs are supported in PDF 1.5 or later. In order to extract content from all layers, the `IgnoreOCM` property should be set to `True`.

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

6.15.1 Label

Property (get): `Boolean Label`

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 no effect.

6.15.2 Level

Property (get): `Long Level`

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.

6.15.3 Name

Property (get): `String Name`

Get the name of the OCG.

6.15.4 Visible

Property (get, set): `Boolean Visible`

Get or set the visibility of the OCG. 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.

Examples:

ID	OCGs	Level	Hierarchy
0	OCG A	0	- OCG A
1	OCG B	0	- OCG B
2	OCG B1	1	- - OCGB1
3	OCG B2	1	- - OCGB2
4	OCG C	-1	hidden: OCG C

ID	OCGs/Labels	Level	Hierarchy
0	OCG A	0	- OCG A
1	Label B	1	- Lavel 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

6.16 PDFObject Interface

This interface represents a basic PDF object. More information on these types of objects can be found in section 3.2 of the PDF Reference 1.7. 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: Indices of dictionary and array objects are 0-based.

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.

6.16.1 Begin, GetNext, End

Property (get): `Long Begin`

Property (get): `Long End`

Method: `GetNext(Long i)`

Applies to dictionaries.

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 in the dictionary.

Example: C#

```
for (int i = dict.Begin; i != dict.End; i = dict.GetNext(i))
{ /* do something */ }
```

6.16.2 BooleanValue

Property (get): `Boolean BooleanValue`

Get the Boolean value of a Boolean object.

6.16.3 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`.

6.16.4 GetElement

Method: `PDFObject GetElement(Long i)`

Applies to arrays. Return a `PDFObject` Interface to the `i`-th value in the dictionary.

6.16.5 `GetEntry`

Method: `PDFObject GetEntry(String Key)`

Applies to dictionaries. Return a `PDFObject` Interface to the value that corresponds to the `Key` in the dictionary.

6.16.6 `GetKey`

Method: `String GetKey(Long i)`

Applies to dictionaries. Return the string representation of the `i`-th key in the dictionary.

See also `GetValue`.

6.16.7 `GetStream`, `StreamMem`

Property (get, set): `Variant StreamMem`

Method: `Long GetStream(String FileName)`

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

6.16.8 `GetValue`

Method: `PDFObject GetValue(Long i)`

Applies to dictionaries. Return a `PDFObject` Interface to the `i`-th value in the dictionary.

See also `GetKey`.

6.16.9 `IntegerValue`

Property (get): `Long IntegerValue`

Get the integer value of a numeric object.

6.16.10 Name

Property (get): `String Name`

Get the character sequence of a name object. The string is zero terminated.

6.16.11 ObjectNumber

Property (get): `Long ObjectNumber`

Applies to indirect objects. Get the object number.

6.16.12 RealValue

Property (get): `Double RealValue`

Get the real value of a numeric object.

6.16.13 Size

Property (get): `Long Size`

Applies to arrays. Get the size of the array.

6.16.14 StringValue

Property (get): `Variant StringValue`

Get the content of a string object as byte array.

6.16.15 Type

Property (get): `TPDFObjectType Type`

Get the type of the object. See `TPDFObjectType`.

6.17 EmbeddedFile Interface

6.17.1 CheckSum

Property (get): `Variant CheckSum`

Get the 16-byte MD5 check sum.

6.17.2 CreationDate

Property (get): `String CreationDate`

Get the creation date.

6.17.3 FileName

Property (get): `String FileName`

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.

6.17.4 MediaType

Property (get): `String MediaType`

Get the media type (MIME type) of the embedded file as a string.

6.17.5 ModDate

Property (get): `String ModDate`

Get the modification date.

6.17.6 Store

Method: `Boolean Store(String Path)`

Store the embedded file to disk.

Parameter:

`Path` `[String]` The file name and path, where the document shall be stored.

Returns:

`True` If the operation competed successfully.

`False` Otherwise.

6.17.7 `StoreInMemory`

Method: `Variant StoreInMemory()`

Store the embedded file in memory.

Returns:

The embedded file as a byte array.

6.18 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).

6.18.1 `TPDFCompliance`

TPDFCompliance Table

TPDFCompliance	
ePDF11	PDF Version 1.1
ePDF12	PDF Version 1.2
ePDF13	PDF Version 1.3
ePDF14	PDF Version 1.4 (corresponds to Acrobat 5)
ePDF15	PDF Version 1.5

TPDFCompliance Table

`ePDF16`	PDF Version 1.6 (corresponds to Acrobat 7)
`ePDF17`	PDF Version 1.7
`ePDFA1a`	PDF/A 1a, ISO 19005-1, Level A compliance
`ePDFA1b`	PDF/A 1b, ISO 19005-1, Level B compliance
`ePDFA2a`	PDF/A 2a, ISO 19005-2, Level A compliance
`ePDFA2b`	PDF/A 2b, ISO 19005-2, Level B compliance
`ePDFA2u`	PDF/A 2u, ISO 19005-2, Level U compliance
`ePDFA3a`	PDF/A 3a, ISO 19005-3, Level A compliance
`ePDFA3b`	PDF/A 3b, ISO 19005-3, Level B compliance
`ePDFA3u`	PDF/A 3u, ISO 19005-3, Level U compliance

Note that only the values listed above are supported.

6.18.2 `TPDFCompression`

TPDFCompression Table

TPDFCompression	Description
`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: Not all image formats/color depths support all compression types.

6.18.3 TPDFContentObject

See also method GetNextObject.

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

6.18.4 TPDFErrorCode

All TPDFErrorCode enumerations start with a prefix, such as 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 an indication of the severity of the error. These are: Success, Error, Warning and Information. In general, an error is returned if an operation could not be completed. A warning is returned if the operation was completed, but problems occurred in the process.

A list of all error codes is available in the C API's header file bseerror.h, the javadoc documentation of com.pdftools.NativeLibrary.ERRORCODE and the .NET documentation of Pdftools.Pdf.PDFErrorCode. Note that only a few are relevant for the 3-Heights™ PDF Extract API, most of which are listed here:

TPDFErrorCode Table

TPDFErrorCode	Description
PDF_S_SUCCESS	The operation was completed successfully.
LIC_E_NOTSET, LIC_E_NOTFOUND, …	Various license management related errors.
PDF_E_FILEOPEN	Failed to open the file.

TPDFErrorCode Table

PDF_E_FILECREATE	Failed to create the file.
PDF_E_PASSWORD	The authentication failed due to a wrong password.
PDF_E_UNKSECHANDLER	The file uses a proprietary security handler, e.g. for a proprietary digital rights management (DRM) system.
PDF_E_XFANEEDSRENDERING	The file contains unrendered XFA form fields, i.e. the file is an XFA and not a PDF file. The XFA (XML Forms Architecture) specification is referenced as an external document to ISO 32'000-1 (PDF 1.7) and has not yet been standardized by ISO. Technically spoken, an XFA form is included as a resource in a shell PDF. The PDF's page content is generated dynamically from the XFA data, which is a complex, non-standardized process. For this reason, XFA is forbidden by the ISO Standards ISO 19'005-2 (PDF/A-2) and ISO 32'000-2 (PDF 2.0) and newer.

6.18.5 TPDFObjectType

This enumeration is used to indicate a syntactical PDF object type as defined in the PDF Reference 1.7, section "3.2 Objects".

Value	Description
eTypeNull	The null PDF object.
eTypeBoolean	A PDF boolean object.
eTypeInteger	A PDF numeric object which represents an integer number.
eTypeReal	A PDF numeric object which represents a real-valued number.
eTypeString	A PDF string object.
eTypeName	A PDF name object.
eTypeArray	An PDF array object.
eTypeDictionary	A PDF dictionary object.
eTypeStream	A PDF stream object.
eTypeIndirect	A PDF indirect object.

6.18.6 TPDFOrientation

TPDFOrientation Table

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.

6.18.7 TPDFTextExtractConfiguration

Value	Description
eTECBreakTextState	Start new text object, if text state changes (font, font size, horizontal scaling). Set this flag, if text state is important to you.
eTECBreakGraphicsState	Start new text object, if graphics state changes (color). Set this flag, if the color is important to you.
eTECBreakSpaceUnicode	Start new text object, if extracted text contains a blank Unicode (\t, non-breaking space, etc.). Do not set this flag, 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 flag, 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 (horizontal displacement), but no more than 10'000 space widths. The merge is done my 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.

7 Version History

Some of the documented changes below may be preceded by a marker that specifies the interface technologies the change applies to. E.g. [C, Java] applies to the C and the Java interface.

7.1 Changes in Version 4.10

- Increased robustness against corrupt input PDF documents.
- [C] **Clarified** Error handling of `TPdfStreamDescriptor` functions.
- Overlapping code ranges in font's ToUnicode tables (used for text extraction) are now supported.

7.2 Changes in Version 4.9

- Improved support for and robustness against corrupt input PDF documents.
- Improved repair of embedded font programs that are corrupt.
- Support OpenType font collections in installed font collection.
- [C] **Changed** return value `pfGetLength` of `TPDFStreamDescriptor` to `pos_t`[5].

Interface Document

- [.NET, C, Java] **New** method `OpenStream()`.

Interface Content

- [.NET, C, COM, Java] **Un-Deprecated** properties `ConvertPathToImage`, `PathImageAntiAlias`, `PathImageBGColor`, and `PathImageResolution`. The "paths as images" feature has been fully re-implemented with the rendering engine R2.

7.3 Changes in Version 4.8

- ToUnicode mappings that map characters to sequences of Unicodes (longer than 1) are now supported. This includes special ligatures and surrogate pairs.
- The space width heuristic has been improved. This algorithm is required in order to estimate the width of a space in fonts that contain no space character. The space width is used to detect word breaks for example.
- The creation of annotation appearances has been optimized to use less memory and processing time.
- Added repair functionality for TrueType font programs whose glyphs are not ordered correctly.

Interface Font

- [.NET, C, COM, Java] **New** property `Font.IsCID`.
- [.NET, C, COM, Java] **New** property `Font.HasUnicodes`.

[5] This has no effect on neither the .NET, Java, nor COM API

Interface Document

- [.NET, C, COM, Java] **New** property `ProductVersion` to identify the product version.
- [.NET] **Deprecated** method `GetLicenseIsValid`.
- [.NET] **New** property `LicenseIsValid`.

8 Licensing, Copyright, and Contact

PDF Tools AG is a world leader in PDF (Portable Document Format) software, delivering reliable PDF products to international customers in all market segments.

PDF Tools AG provides server-based software products designed specifically for developers, integrators, consultants, customizing specialists and IT-departments. Thousands of companies worldwide use our products directly and hundreds of thousands of users benefit from the technology indirectly via a global network of OEM partners. The tools can be easily embedded into application programs and are available for a multitude of operating system platforms.

Licensing and Copyright

Contact

PDF Tools AG
Kasernenstrasse 1
8184 Bachenbülach
Switzerland
http://www.pdf-tools.com
pdfsales@pdf-tools.com