3-Heights™
PDF Extract API

Version 6.12.0
Contents

1 Introduction ............................................................................................................. 8
  1.1 Description ........................................................................................................ 8
  1.2 Functions .......................................................................................................... 8
  1.2.1 Features ...................................................................................................... 8
  1.2.2 Formats ........................................................................................................ 9
  1.2.3 Conformance ................................................................................................ 9
  1.3 Interfaces .......................................................................................................... 10
  1.4 Operating Systems ............................................................................................ 10
  1.5 How to Best Read this Manual ........................................................................ 10

2 Installation and Deployment .................................................................................... 11
  2.1 Windows ............................................................................................................ 11
  2.2 Linux and macOS ............................................................................................. 12
    2.2.1 Linux ......................................................................................................... 12
    2.2.2 macOS ...................................................................................................... 12
  2.3 Zip Archive ....................................................................................................... 13
    2.3.1 Development ............................................................................................. 13
    2.3.2 Deployment ............................................................................................... 14
  2.4 NuGet Package ................................................................................................ 15
  2.5 Interface Specific Installation Steps ................................................................... 16
    2.5.1 COM Interface ......................................................................................... 16
    2.5.2 Java Interface ........................................................................................... 16
    2.5.3 .NET Interface ......................................................................................... 17
    2.5.4 C Interface ............................................................................................... 17
  2.6 Uninstall, Install a New Version ........................................................................ 17
  2.7 Color Profiles ................................................................................................... 17
    2.7.1 Default Color Profiles ............................................................................... 18
    2.7.2 Get Other Color Profiles ........................................................................... 18

3 License Management ................................................................................................. 19

4 Programming Interfaces .......................................................................................... 20
  4.1 Visual Basic 6 ................................................................................................. 20
  4.2 ASP Script ........................................................................................................ 20
  4.3 .NET ................................................................................................................ 21
    4.3.1 Visual Basic .............................................................................................. 22
    4.3.2 C# ............................................................................................................. 23
    4.3.3 Deployment ............................................................................................... 23
    4.3.4 Troubleshooting: TypeInitializationException ........................................ 23

5 User's Guide .............................................................................................................. 25
  5.1 Text Extraction ................................................................................................. 25
    5.1.1 Undesired/Missing Blanks ....................................................................... 25
    5.1.2 Extracted Text is Unreadable .................................................................... 25
    5.1.3 Handling of Symbolic and Non-Symbolic Fonts ....................................... 25
    5.1.4 Text Extraction of Text Marked as Symbolic ............................................ 26
  5.2 Image Extraction ............................................................................................... 26
    5.2.1 Image Resolution ....................................................................................... 26
    5.2.2 Image Orientation ...................................................................................... 26
  5.3 Optional Content (Layers) ................................................................................ 27
6 Interface Reference

6.1 Document Interface

6.1.1 Author

6.1.2 Close

6.1.3 Compliance

6.1.4 CreationDate

6.1.5 Creator

6.1.6 GetDestination

6.1.7 DPartRoot

6.1.8 GetFirstChildColorSpaceResource, GetNextChildColorSpaceResource

6.1.9 GetFirstChildEmbeddedFile, GetNextChildEmbeddedFile

6.1.10 GetFirstChildFontResource, GetNextChildFontResource

6.1.11 GetFirstChildImageResource, GetNextChildImageResource

6.1.12 GetFirstChildOutlineItem, GetNextChildOutlineItem

6.1.13 GetInfoEntry

6.1.14 GetPDFObject

6.1.15 GetOcg, OcgCount

6.1.16 GetPageLabel

6.1.17 GetXMPMetadata

6.1.18 GetXMPMetadataMem

6.1.19 IsCollection

6.1.20 IsEncrypted

6.1.21 IsLinearized

6.1.22 Keywords

6.1.23 LastError

6.1.24 LastErrorMessage

6.1.25 LicenseIsValid

6.1.26 MajorVersion, MinorVersion

6.1.27 ModifyDate

6.1.28 Open

6.1.29 OpenMem

6.1.30 OpenStream

6.1.31 Page

6.1.32 PageCount

6.1.33 PageNo

6.1.34 Producer

6.1.35 ProductVersion

6.1.36 SetLicenseKey

6.1.37 Subject

6.1.38 Title

6.2 Page Interface

6.2.1 ArtBox

6.2.2 BleedBox

6.2.3 Content

6.2.4 CropBox

6.2.5 DeviceColorant

6.2.6 Document

6.2.7 GetFirstChildAnnotations, GetNextAnnotation

6.2.8 HasColor

6.2.9 MediaBox

6.2.10 Rotate

6.2.11 TrimBox
6.3  Content Interface ................................................................. 40
  6.3.1  BreakWords .......................................................... 41
  6.3.2  BoundingBox ......................................................... 41
  6.3.3  ConvertPathToImage .............................................. 41
  6.3.4  ExpandLigatures ................................................... 41
  6.3.5  Flags ................................................................. 42
  6.3.6  GetNextImage ....................................................... 42
  6.3.7  GetNextObject ...................................................... 42
  6.3.8  GetNextPath ........................................................ 43
  6.3.9  GetNextText ........................................................ 43
  6.3.10  GraphicsState ...................................................... 43
  6.3.11  IgnoreOCM ........................................................ 44
  6.3.12  Image ............................................................... 44
  6.3.13  OCM ................................................................. 44
  6.3.14  Path ................................................................. 44
  6.3.15  PathImageAntiAlias .............................................. 45
  6.3.16  PathImageBGColor ............................................... 45
  6.3.17  PathImageResolution ............................................ 46
  6.3.18  Reset .................................................. 46
  6.3.19  SpaceFactor ...................................................... 46
  6.3.20  Text ................................................................. 46
  6.3.21  TextExtConfiguration ......................................... 47
  6.3.22  TranslateSymbolic ............................................... 47
6.4  Image Interface ................................................................. 47
  6.4.1  Alternates .......................................................... 47
  6.4.2  BitsPerComponent .................................................. 48
  6.4.3  ChangeOrientation ................................................. 48
  6.4.4  ColorSpace ........................................................ 48
  6.4.5  Compression ......................................................... 48
  6.4.6  ConvertToRGB ........................................................ 48
  6.4.7  GetImage ........................................................... 49
  6.4.8  GetResolution ....................................................... 49
  6.4.9  Height ............................................................... 49
  6.4.10  IsBitonal .......................................................... 49
  6.4.11  IsColor ............................................................. 49
  6.4.12  IsMonochrome ..................................................... 50
  6.4.13  Mask ............................................................... 50
  6.4.14  ObjNumber ......................................................... 50
  6.4.15  Samples ............................................................ 50
  6.4.16  SMask ............................................................... 50
  6.4.17  Store ............................................................... 51
  6.4.18  StoreInMemory ................................................... 51
  6.4.19  Width .............................................................. 52
6.5  Text Interface ..................................................................... 52
  6.5.1  BoundingBox .......................................................... 52
  6.5.2  FontWeight .......................................................... 53
  6.5.3  Length ................................................................. 53
  6.5.4  QuadPoints .......................................................... 53
  6.5.5  RawString ........................................................... 53
  6.5.6  Rotation ............................................................... 54
  6.5.7  StringLength ........................................................ 54
  6.5.8  UnicodeString ..................................................... 54

© PDF Tools AG – Premium PDF Technology
3-Heights™ PDF Extract API, October 21, 2020 | 3/92
6.5.9 TextMatrix ................................................................. 54
6.5.10 Width ........................................................................ 54
6.5.11 XPos, YPos, Position .......................................................... 55
6.6 GraphicsState Interface ............................................................ 55
6.6.1 AlphaIsShape ................................................................. 55
6.6.2 BlendMode .................................................................... 55
6.6.3 CharSpacing ................................................................. 56
6.6.4 CTM ........................................................................ 56
6.6.5 DashArray ..................................................................... 56
6.6.6 DashPhase ..................................................................... 56
6.6.7 FillAlphaConstant ............................................................ 56
6.6.8 FillColorCMYK ............................................................... 56
6.6.9 FillColorRGB ................................................................. 57
6.6.10 FillColorSpace ............................................................... 58
6.6.11 FillOverprintFlag ............................................................. 58
6.6.12 FlatnessTolerance ............................................................ 58
6.6.13 Font ........................................................................ 58
6.6.14 FontSize ...................................................................... 58
6.6.15 HorizontalScaling ............................................................. 58
6.6.16 Leading ...................................................................... 59
6.6.17 LineCap ...................................................................... 59
6.6.18 LineJoin ...................................................................... 59
6.6.19 LineWidth .................................................................... 59
6.6.20 MiterLimit ................................................................... 59
6.6.21 OverprintMode ............................................................... 60
6.6.22 RenderingIntent ............................................................. 60
6.6.23 SmoothnessTolerance ....................................................... 60
6.6.24 SpaceWidth ................................................................. 60
6.6.25 StrokeAdjustment ............................................................ 60
6.6.26 StrokeAlphaConstant ....................................................... 61
6.6.27 StrokeColorCMYK .......................................................... 61
6.6.28 StrokeColorRGB ............................................................ 61
6.6.29 StrokeColorSpace ........................................................... 61
6.6.30 StrokeOverprintFlag ........................................................ 61
6.6.31 TextKnockout ................................................................. 62
6.6.32 TextRenderingMode ......................................................... 62
6.6.33 TextRise ...................................................................... 62
6.6.34 WordSpacing ................................................................. 62
6.7 Font Interface ................................................................. 62
6.7.1 Ascent ...................................................................... 62
6.7.2 AvgWidth .................................................................... 63
6.7.3 BaseName .................................................................... 63
6.7.4 CapHeight .................................................................... 63
6.7.5 CharSet ...................................................................... 63
6.7.6 Descent ....................................................................... 63
6.7.7 Encoding ..................................................................... 63
6.7.8 Flags ........................................................................ 64
6.7.9 FontBoundingBox .............................................................. 64
6.7.10 FontFile ..................................................................... 64
6.7.11 FontFileType ................................................................. 65
6.7.12 HasUnicodes ................................................................. 65
6.7.13 IsCID ...................................................................... 65
6.7.14  ItalicAngle  ................................................................. 65
6.7.15  Leading ................................................................. 65
6.7.16  MaxWidth ............................................................... 66
6.7.17  MissingWidth ........................................................... 66
6.7.18  StemH, StemV ............................................................ 66
6.7.19  Type ................................................................. 66
6.7.20  Widths ................................................................. 66
6.7.21  WMode ................................................................. 66
6.7.22  XHeight ................................................................. 67
6.8  ColorSpace Interface ..................................................... 67
6.8.1  BaseColorSpace ......................................................... 67
6.8.2  ColorantName ........................................................... 67
6.8.3  ComponentsPerPixel .................................................. 67
6.8.4  HighIndex ............................................................. 68
6.8.5  IsColor ................................................................. 68
6.8.6  IsIndexed ............................................................... 68
6.8.7  IsMonochrome .......................................................... 68
6.8.8  Lookup ................................................................. 68
6.8.9  Name ................................................................. 68
6.9  TransformMatrix Interface ............................................. 69
6.9.1  a, b, c, d, e, f ......................................................... 69
6.9.2  Orientation ............................................................ 69
6.9.3  Rotation ............................................................... 69
6.9.4  XScaling, YScaling ..................................................... 70
6.9.5  XSkeW, YSkew ........................................................ 70
6.9.6  XTranslation, YTranslation ........................................ 70
6.10  AlternateImage Interface .............................................. 70
6.10.1  DefaultForPrinting ................................................... 70
6.10.2  Image ................................................................. 70
6.11  AlternateArray Interface .............................................. 70
6.11.1  Count ................................................................. 71
6.11.2  GetElement ........................................................... 71
6.12  Annotation Interface .................................................. 71
6.12.1  AttachedFile ........................................................ 71
6.12.2  Color ................................................................. 71
6.12.3  Contents .............................................................. 71
6.12.4  Date ................................................................. 72
6.12.5  Dest ................................................................. 72
6.12.6  Flags ................................................................. 72
6.12.7  IsMarkup ............................................................. 72
6.12.8  Name ................................................................. 73
6.12.9  Rect ................................................................. 73
6.12.10  Subj ................................................................. 73
6.12.11  Subtype .............................................................. 73
6.12.12  TextLabel ........................................................... 73
6.12.13  QuadPoints .......................................................... 73
6.12.14  URI ................................................................. 74
6.12.15  Vertices ............................................................. 74
6.13  OutlineItem Interface ............................................... 74
6.13.1  Count ................................................................. 74
6.13.2  Dest ................................................................. 74
6.13.3  Title ................................................................. 74
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.14</td>
<td>Destination Interface</td>
</tr>
<tr>
<td>6.14.1</td>
<td>Bottom</td>
</tr>
<tr>
<td>6.14.2</td>
<td>Left</td>
</tr>
<tr>
<td>6.14.3</td>
<td>PageNo</td>
</tr>
<tr>
<td>6.14.4</td>
<td>Right</td>
</tr>
<tr>
<td>6.14.5</td>
<td>Top</td>
</tr>
<tr>
<td>6.14.6</td>
<td>Type</td>
</tr>
<tr>
<td>6.14.7</td>
<td>Zoom</td>
</tr>
<tr>
<td>6.15</td>
<td>Ocg Interface</td>
</tr>
<tr>
<td>6.15.1</td>
<td>Label</td>
</tr>
<tr>
<td>6.15.2</td>
<td>Level</td>
</tr>
<tr>
<td>6.15.3</td>
<td>Name</td>
</tr>
<tr>
<td>6.15.4</td>
<td>Visible</td>
</tr>
<tr>
<td>6.16</td>
<td>PDFObject Interface</td>
</tr>
<tr>
<td>6.16.1</td>
<td>Begin, GetNext, End</td>
</tr>
<tr>
<td>6.16.2</td>
<td>BooleanValue</td>
</tr>
<tr>
<td>6.16.3</td>
<td>Dispose, DestroyObject</td>
</tr>
<tr>
<td>6.16.4</td>
<td>GetElement</td>
</tr>
<tr>
<td>6.16.5</td>
<td>GetEntry</td>
</tr>
<tr>
<td>6.16.6</td>
<td>GetKey</td>
</tr>
<tr>
<td>6.16.7</td>
<td>GetStream, StreamMem</td>
</tr>
<tr>
<td>6.16.8</td>
<td>GetValue</td>
</tr>
<tr>
<td>6.16.9</td>
<td>IntegerValue</td>
</tr>
<tr>
<td>6.16.10</td>
<td>Name</td>
</tr>
<tr>
<td>6.16.11</td>
<td>ObjectNumber</td>
</tr>
<tr>
<td>6.16.12</td>
<td>RealValue</td>
</tr>
<tr>
<td>6.16.13</td>
<td>Size</td>
</tr>
<tr>
<td>6.16.14</td>
<td>StringValue</td>
</tr>
<tr>
<td>6.16.15</td>
<td>TextStringValue</td>
</tr>
<tr>
<td>6.16.16</td>
<td>Type</td>
</tr>
<tr>
<td>6.17</td>
<td>EmbeddedFile Interface</td>
</tr>
<tr>
<td>6.17.1</td>
<td>CheckSum</td>
</tr>
<tr>
<td>6.17.2</td>
<td>CreationDate</td>
</tr>
<tr>
<td>6.17.3</td>
<td>FileName</td>
</tr>
<tr>
<td>6.17.4</td>
<td>MediaType</td>
</tr>
<tr>
<td>6.17.5</td>
<td>ModDate</td>
</tr>
<tr>
<td>6.17.6</td>
<td>Store</td>
</tr>
<tr>
<td>6.17.7</td>
<td>StoreInMemory</td>
</tr>
<tr>
<td>6.18</td>
<td>DPartRoot Interface</td>
</tr>
<tr>
<td>6.18.1</td>
<td>NodeNameCount, NodeName</td>
</tr>
<tr>
<td>6.18.2</td>
<td>RecordLevel</td>
</tr>
<tr>
<td>6.18.3</td>
<td>RootNode</td>
</tr>
<tr>
<td>6.19</td>
<td>DPartNode Interface</td>
</tr>
<tr>
<td>6.19.1</td>
<td>ChildrenCount, Child</td>
</tr>
<tr>
<td>6.19.2</td>
<td>DPM</td>
</tr>
<tr>
<td>6.19.3</td>
<td>PageRange</td>
</tr>
<tr>
<td>6.20</td>
<td>Enumerations</td>
</tr>
<tr>
<td>6.20.1</td>
<td>TPDFCompliance Enumeration</td>
</tr>
<tr>
<td>6.20.2</td>
<td>TPDFCompression Enumeration</td>
</tr>
<tr>
<td>6.20.3</td>
<td>TPDFContentObject Enumeration</td>
</tr>
<tr>
<td>6.20.4</td>
<td>TPDFErrorCode Enumeration</td>
</tr>
<tr>
<td>6.20.5</td>
<td>TPDFObject Type Enumeration</td>
</tr>
</tbody>
</table>
6.20.6  TPDFOrientation Enumeration ................................................................. 87
6.20.7  TPDFTextExtractConfiguration Enumeration .......................................... 87

7  Version History .................................................................................................. 89
  7.1  Changes in Version 6 ...................................................................................... 89
  7.2  Changes in Version 5 ...................................................................................... 89
  7.3  Changes in Version 4.12 .................................................................................. 89
  7.4  Changes in Version 4.11 .................................................................................. 89
  7.5  Changes in Version 4.10 .................................................................................. 90
  7.6  Changes in Version 4.9 .................................................................................... 90
  7.7  Changes in Version 4.8 .................................................................................... 91

8  Licensing, Copyright, and Contact ................................................................. 92
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
- Extract document parts for PDF/X or PDF 2.0.
- 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 (PDF 1.0, . . . , PDF 1.7)
- PDF 2.0
- PDF/A-1, PDF/A-2, PDF/A-3

### 1.2.3 Conformance

**Standards:**
- ISO 32000-1 (PDF 1.7)
- ISO 32000-2 (PDF 2.0)
- ISO 19005-1 (PDF/A-1)
- ISO 19005-2 (PDF/A-2)
- ISO 19005-3 (PDF/A-3)
1.3 Interfaces

The following interfaces are available:

- C
- Java
- .NET Framework
- .NET Core\(^1\)
- COM

1.4 Operating Systems

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

- Windows Client 7+ | x86 and x64
- Linux:
  - Red Hat, CentOS, Oracle Linux 7+ | x64
  - Fedora 29+ | x64
  - Debian 8+ | x64
  - Other: Linux kernel 2.6+, GCC toolset 4.8+ | x64
- macOS 10.10+ | x64

\(^1\) indicates the minimum supported version.

1.5 How to Best Read this Manual

If you are reading this manual for the first time, i.e. would like to evaluate the software, the following steps are suggested.

1. Read the chapter Introduction to verify this product meets your requirements.
2. Identify what interface your programming language uses.
3. Read and follow the instructions in the chapter Installation and Deployment.
4. In the chapter Programming Interfaces find your programming language. Please note that not every language is covered in this manual.
   For most programming languages there is sample code available. For a start it is generally best to refer to these samples rather than writing code from scratch.
5. (Optional) Read the chapter User's Guide for general information about the API. Read the Interface Reference for specific information about the functions of the API.

\(^1\) Limited supported OS versions. Operating Systems
2 Installation and Deployment

2.1 Windows

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

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](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. A different version can be selected using the combo box.

The product comes as a [Zip Archive](zip-archive) containing all files, or as a [NuGet Package](nuget-package) containing all files for development in .NET.

There is a 32 and a 64-bit version 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 archive as well as the NuGet package contain both the 32-bit and the 64-bit version of the product.

3. If you are using the ZIP archive, do the following. Unzip the archive to a local folder, e.g. `C:\Program Files\PDF Tools AG`.

This creates the following subdirectories (see also [Zip Archive](zip-archive)):

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

4. The usage of the NuGet package is described in section [NuGet Package](nuget-package).

5. (Optional) Register your license key using the [License Management](license-management).

6. Identify which interface you are using. Perform the specific installation steps for that interface described in [Interface Specific Installation Steps](interface-specific-installation-steps).

7. Make sure your platform meets the requirements regarding color spaces described in chapter [Color Profiles](color-profiles).

2.2 Linux and macOS

This section describes installation steps required on Linux or macOS.

The Linux and macOS 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

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bin/x64/libPdfParser.so</td>
<td>This is the shared library that contains the main functionality. The file's</td>
</tr>
<tr>
<td></td>
<td>extension differs on macOS (.dylib instead of .so).</td>
</tr>
<tr>
<td>doc/<em>.</em></td>
<td>Documentation</td>
</tr>
<tr>
<td>include/<em>.</em></td>
<td>Contains header files to include in your C/C++ project.</td>
</tr>
<tr>
<td>jar/EXPA.jar</td>
<td>Java API archive.</td>
</tr>
<tr>
<td>samples</td>
<td>Example code.</td>
</tr>
</tbody>
</table>

#### 2.2.1 Linux

1. Unpack the archive in an installation directory, e.g. `/opt/pdf-tools.com/`
2. Verify that the GNU shared libraries required by the product are available on your system:

   ```
   ldd libPdfParser.so
   ```

   In case the above reports any missing libraries you have three options:
   a. Download an archive that is linked to a different version of the GNU shared libraries and verify whether they are available on your system. Use any version whose requirements are met. Note that this option is not available for all platforms.
   b. Use your system's package manager to install the missing libraries. It usually suffices to install the package `libstdc++6`.
   c. Use GNU shared libraries provided by PDF Tools AG:
      2. Download the GNU shared libraries for your platform.
      3. Install the libraries manually according your system's documentation. This typically involves copying them to your library directory, e.g. `/usr/lib` or `/usr/lib64`, and running `ldconfig`.
      4. Verify that the GNU shared libraries required by the product are available on your system now.
   3. Create a link to the shared library from one of the standard library directories, e.g:

      ```
      ln -s /opt/pdf-tools.com/bin/x64/libPdfParser.so /usr/lib
      ```

   4. Optionally register your license key using the license manager.
   5. Identify which interface you are using. Perform the specific installation steps for that interface described in 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 Zip Archive

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.

<table>
<thead>
<tr>
<th>Interface</th>
<th>Programming Languages</th>
</tr>
</thead>
<tbody>
<tr>
<td>.NET</td>
<td>The MS software platform .NET can be used with any .NET capable programming language such as:</td>
</tr>
<tr>
<td></td>
<td>- C#</td>
</tr>
<tr>
<td></td>
<td>- VB .NET</td>
</tr>
<tr>
<td></td>
<td>- J#</td>
</tr>
<tr>
<td></td>
<td>- others</td>
</tr>
<tr>
<td></td>
<td>For a convenient way to use this interface, see NuGet Package.</td>
</tr>
<tr>
<td>Java</td>
<td>The Java interface is available on all platforms.</td>
</tr>
<tr>
<td>COM</td>
<td>The component object model (COM) interface can be used with any COM-capable programming language, such as:</td>
</tr>
<tr>
<td></td>
<td>- MS Visual Basic</td>
</tr>
<tr>
<td></td>
<td>- MS Office Products such as Access or Excel (VBA)</td>
</tr>
<tr>
<td></td>
<td>- C++</td>
</tr>
<tr>
<td></td>
<td>- VBScript</td>
</tr>
<tr>
<td></td>
<td>- others</td>
</tr>
<tr>
<td></td>
<td>This interface is available in the Windows version only.</td>
</tr>
<tr>
<td>C</td>
<td>The native C interface is for use with C and C++. This interface is available on all platforms.</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Files for Development</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>bin&lt;platform&gt;\PDFParser.dll</td>
</tr>
<tr>
<td>bin*NET.dll</td>
</tr>
<tr>
<td>bin*NET.xml</td>
</tr>
</tbody>
</table>
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 Development

<table>
<thead>
<tr>
<th>Name</th>
<th>.NET</th>
<th>Java</th>
<th>COM</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc\PDFParser.idl</td>
<td>Doc.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>doc\javadoc*.*</td>
<td>Doc.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>include\expa_c.h</td>
<td>Req.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>include*.*</td>
<td>Opt.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>jar\EXPA.jar</td>
<td>Req.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lib&lt;platform&gt;\PDFParser.lib</td>
<td>Req.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

### File Description

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bin&lt;platform&gt;\PDFParser.dll</td>
<td>This is the DLL that contains the main functionality (required), where &lt;platform&gt; is either Win32 or x64 for the 23-bit or the 64-bit library respectively.</td>
</tr>
<tr>
<td>bin*NET.dll</td>
<td>The .NET assemblies are required when using the .NET interface. The files bin*NET.xml contain the corresponding XML documentation for MS Visual Studio.</td>
</tr>
<tr>
<td>doc*.*</td>
<td>Various documentations.</td>
</tr>
<tr>
<td>include*.*</td>
<td>Contains files to include in your C / C++ project.</td>
</tr>
<tr>
<td>lib&lt;platform&gt;\PDFParser.lib</td>
<td>On Windows operating systems, the object file library needs to be linked to the C/C++ project.</td>
</tr>
<tr>
<td>jar\EXPA.jar</td>
<td>The Java API archive.</td>
</tr>
<tr>
<td>samples*.*</td>
<td>Contains sample programs in different programming languages.</td>
</tr>
</tbody>
</table>

---

2 Not required for Linux or macOS.

3 These files must reside in the same directory as PDFParser.dll.
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 NuGet Package

Nuget is a package manager that facilitates the integration of libraries for the software development in .NET. The nuget package for the 3-Heights™ PDF Extract API contains all the libraries needed, managed and native.

**Installation**  

In Visual Studio click on “Tools” and then “Options”. Select “NuGet Package Manager” and add the location of the downloaded package in “Package Sources”.

Right-click on a .NET project in Visual Studio and select “Manage NuGet Packages...”. Finally, select the package source that was defined above and browse to the desired package.

**Development**  

The required native libraries are loaded automatically. All project platforms are supported, including “AnyCPU”.

In order to use the software, you must first install a license key for the 3-Heights™ PDF Extract API. To do this you have to download the product kit and use the license manager in it. See also License Management.
2.5 Interface Specific Installation Steps

2.5.1 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. 4

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.

2.5.2 Java Interface

The 3-Heights™ PDF Extract API requires Java version 7 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" sampleApplication.java
```

For execution Additionally the library PDFParser.dll needs be in one of the system's library directories 5 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 on Windows) depending on the platform of the Java VM 6.

4 Otherwise you get the following message: `LoadLibrary("PDFParser.dll") failed - The specified module could not be found`.

5 On Windows defined by the environment variable PATH and e.g. on Linux defined by LD_LIBRARY_PATH.

6 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”
```java
java -classpath ".;C:\Program Files\PDF Tools AG\EXPA.jar" ^
"-Djava.library.path=C:\Program Files\PDF Tools AG\bin\x64" sampleApplication
```

Note that on Linux or macOS, the path separator usually is a colon and hence the above changes to something like:

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

### 2.5.3 .NET Interface

The 3-Heights™ PDF Extract API does not provide a pure .NET solution. Instead, it consists of a native library and .NET assemblies, which call the native library. This has to be accounted for when installing and deploying the tool.

It is recommended to use the [NuGet Package](https://www.nuget.org/packages). This ensures the correct handling of both the .NET assemblies and the native library.

Alternatively, the files in the [Zip Archive](https://www.pdf-tools.com/products/pdf-extract-api/) can be used directly in a Visual Studio project targeting .NET Framework 2.0 or later. To achieve this, proceed as follows.

The .NET assemblies (*.NET.dll) are to be added as references to the project; They are needed at compile time. PDFParser.dll is not a .NET assembly, but a native library. It is not to be added as a reference to the project. Instead, it is loaded during execution of the application.

For the operating system to find and successfully load the native library PDFParser.dll, it must match the executing application's bitness (32-bit versus 64-bit) and it must reside in either of the following directories:

- In the same directory as the application that uses the library.
- In a subdirectory `win-x86` or `win-x64` for 32-bit or 64-bit applications respectively.
- In a directory that is listed in the PATH environment variable.

In Visual Studio, when using the platforms “x86” or “x64”, the above can be achieved by adding the 32-bit or 64-bit PDFParser.dll respectively as an “existing item” to the project, and setting its property “Copy to output directory” to true. When using the “AnyCPU” platform, then you have to make sure by some other means that both the 32-bit and the 64-bit PDFParser.dll are copied to subdirectories `win-x86` and `win-x64` of the output directory respectively.

### 2.5.4 C Interface

- The header file `expa_c.h` needs to be included in the C/C++ program.
- On Windows operating systems, 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%`).

### 2.6 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.

### 2.7 Color Profiles

When extracting images, a color conversion might be necessary.
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 appropriate 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.

2.7.1 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%\System32\spool\drivers\color
2. directory Icc, which must be a direct sub-directory of where the PDFParser.dll resides.

**Linux and macOS**

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

2.7.2 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.color.org/srgbprofiles.html](http://www.color.org/srgbprofiles.html)
3 License Management

The 3-Heights™ PDF Extract API requires a valid license in order to run correctly. If no license key is set or the license is not valid, then most of the interface elements documented in Interface Reference will fail with an error code and error message indicating the reason.

More information about license management is available in the license key technote.
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.

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.

```vba
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:
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:

```vbscript
<%@ Language=VBScript %>
<%
    option explicit
    dim pdfDoc
    if not pdfDoc.Open(Server.MapPath("myfile.pdf")) then
        Response.Write "<p>
            Could not open file." & "<br>
        end if
        Response.Write "<p>
            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:
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:

![Screenshot of Visual Basic project settings]

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

**Example:**

```vbnet
document.Open(...)
document.PageNo = 1
Dim content = document.Page.Content
...```

4.3.2 C#

3. Add the following namespaces:

**Example:**

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

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

**Example:**

```csharp
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. See also .NET Interface.
2. For deployment, all items in the project's output directory (e.g. bin\Release) must be copied to the target computer. This includes the 3-Heights™ PDF Extract API's .NET assemblies (*.NET.dll) as well as the native library (PDFParser.dll) in its 32 bit or 64 bit version or both. The native library can alternatively be copied to a directory listed in the PATH environment variable, e.g. %SystemRoot%\System32.
3. It is crucial, that the native library PDFParser.dll is found at execution time, and that the native library's format (32 bit versus 64 bit) matches the operating system.
4. The output directory may contain multiple versions of the native library, e.g. for Windows 32 bit, Windows 64 bit, MacOS 64 bit, and Linux 64 bit. Only the versions that match the target computer's operating system need be deployed.
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:

- using the [NuGet Package](#).
- 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
- manually 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 in the [Zip Archive](#): 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.

(Using the [NuGet Package](#) normally ensures that the matching native DLL is loaded at execution time.)

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, then a different native DLL has to be used, depending on the Windows platform. 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. The latter can be achieved by placing both the 32 bit and the 64 bit native DLL in subdirectories `win-x86` and `win-x64` of the application's directory respectively.

- **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.

- **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.
- In case the font file is not embedded: The standard encoding is used.

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`, `OcgCount` method.

As an example, look at the document [www.pdf-tools.com/public/downloads/samples/layers.pdf](http://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:

<table>
<thead>
<tr>
<th>ID</th>
<th>name</th>
<th>level</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Red</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>Green</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>Blue</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>Gray</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>Gray 64</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>Gray 128</td>
<td>1</td>
</tr>
</tbody>
</table>

#### Extracting objects from layers.pdf:

<table>
<thead>
<tr>
<th>Type</th>
<th>Property = Value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>BeginOCM</td>
<td>OCM=&quot;0&quot;</td>
<td>Visibility of subsequent objects is defined by the state of OCG 0 (“Red”)</td>
</tr>
<tr>
<td>..Path</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EndOCM</td>
<td></td>
<td>End of OCM segment</td>
</tr>
<tr>
<td>BeginOCM</td>
<td>OCM=&quot;1&quot;</td>
<td>OCG 1 is “Green”</td>
</tr>
<tr>
<td>..Path</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EndOCM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BeginOCM</td>
<td>OCM=&quot;2&quot;</td>
<td>OCG 2 is “Blue”</td>
</tr>
<tr>
<td>..Path</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EndOCM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BeginOCM</td>
<td>OCM=&quot;1 &amp;&amp; 2&quot;</td>
<td>Subsequent objects are visible, if OCG 1 and OCG 2 are ON.</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>Path</td>
<td>Path=yellow square</td>
<td></td>
</tr>
<tr>
<td>EndOCM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BeginOCM</td>
<td>OCM=&quot;3&quot;</td>
<td>OCG 3 is “Gray”, parent OCG of 4 and 5</td>
</tr>
<tr>
<td>BeginOCM</td>
<td>OCM=&quot;4&quot;</td>
<td>Note that OCM blocks can be nested, typically uses for hierarchical OCGs</td>
</tr>
<tr>
<td>Path</td>
<td>Path=gray 64 square</td>
<td></td>
</tr>
<tr>
<td>EndOCM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BeginOCM</td>
<td>OCM=&quot;5&quot;</td>
<td>OCG 5 is “Gray 128”</td>
</tr>
<tr>
<td>Path</td>
<td>Path=gray 128 square</td>
<td></td>
</tr>
<tr>
<td>EndOCM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EndOCM</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
6 Interface Reference

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 conformance 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` object if the specified destination exists.
- `Nothing` otherwise.

**Parameter:**
- `Destination` [String] The named destination

### 6.1.7 DPartRoot

**Property (get):** `DPartRoot DPartRoot`

Get a `DPartRoot` object that represents the document parts' root element of a PDF 2.0 or a PDF/VT document. If no document part information is present then this property returns `Nothing`.

### 6.1.8 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` object for the next color space resource if there is any.
- `Nothing` otherwise.

### 6.1.9 GetFirstEmbeddedFile, GetNextEmbeddedFile

**Method:** `EmbeddedFile GetFirstEmbeddedFile()`

**Method:** `EmbeddedFile GetNextEmbeddedFile()`
Use \texttt{GetFirstEmbeddedFile} to get the first embedded file. After this you can use \texttt{GetNextEmbeddedFile} to get further embedded files. Embedded files of both the document's collection (PDF Portfolio) and of FileAttachment annotations are returned.

\textbf{Returns:}

- An \texttt{EmbeddedFile} object for the next embedded file if there is any.
- \texttt{Nothing} otherwise.

\textbf{6.1.10 GetFirstFontResource, GetNextFontResource}

\textbf{Method:} Font \texttt{GetFirstFontResource()}
\textbf{Method:} Font \texttt{GetNextFontResource()}

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

\textbf{Returns:}

- A \texttt{Font} object for the next font resource if there is any.
- \texttt{Nothing} otherwise.

\textbf{6.1.11 GetFirstImageResource, GetNextImageResource}

\textbf{Method:} Image \texttt{GetFirstImageResource()}
\textbf{Method:} Image \texttt{GetNextImageResource()}

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

\textbf{6.1.12 GetFirstOutlineItem, GetNextOutlineItem}

\textbf{Method:} OutlineItem \texttt{GetFirstOutlineItem()}
\textbf{Method:} OutlineItem \texttt{GetNextOutlineItem(Long MaxLevel, Boolean ReturnOpenOnly)}
\textbf{Method:} Long \texttt{GetCurrentOutlineLevel()}

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

\textbf{Parameters:}

- \texttt{MaxLevel} \ [Long] (optional, default 20) The maximum level of the depth of the outlines.
- \texttt{ReturnOpenOnly} \ [Boolean] (optional, default \texttt{False}) Return only outlines which are opened.
Returns:
- An `OutlineItem` object for the next outline item if there is any.
- `Nothing` otherwise.

### 6.1.13 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.

### 6.1.14 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"
Returns:

A PDFObject interface to the PDF object specified by the Path string.

### 6.1.15 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:**

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

**Returns:**

A Ocg object for the requested OCG.

### 6.1.16 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:**

```csharp
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.

### 6.1.17 GetXMPMetadata

**Method:** Boolean GetXMPMetadata(String FileName)

© PDF Tools AG – Premium PDF Technology
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.

6.1.18 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.

6.1.19 IsCollection

**Property (get):** Boolean IsCollection

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

6.1.20 IsEncrypted

**Property (get):** Boolean IsEncrypted

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

6.1.21 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.
6.1.22 Keywords

Property (get): String Keywords

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

6.1.23 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.

6.1.24 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.

6.1.25 LicenseIsValid

Property (get): Boolean LicenseIsValid

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

6.1.26 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.

6.1.27 ModDate

Property (get): Date ModDate

Get the modification date of the info object of the document.
6.1.28 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.

6.1.29 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.

6.1.30 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.

### 6.1.31 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.

### 6.1.32 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.

### 6.1.33 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.
6.1.34 Producer

**Property (get):** String **Producer**

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

6.1.35 ProductVersion

**Property (get):** String **ProductVersion**

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

6.1.36 SetLicenseKey

**Method:** Boolean **SetLicenseKey(String LicenseKey)**

Set the license key.

6.1.37 Subject

**Property (get):** String **Subject**

Get the subject from the document's info object.

6.1.38 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 object for 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 object 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` object for the next annotation if there is any.
- `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 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 object for the next image 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 was found, a Text 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 object for the next text if there is any on this page.
- Nothing otherwise.

### 6.3.10 GraphicsState

**Property (get):** GraphicsState GraphicsState
Get the current `GraphicsState` object.
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 a 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, OcgCount` 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: \(\langle x \rangle \langle y \rangle \ m\)
- Line from current point to: \(\langle x \rangle \langle y \rangle \ l\)
- Rectangle: \(\langle x \rangle \langle y \rangle \langle w \rangle \langle h \rangle \ re\)
- Cubic Bezier curve from current point to: \(\langle x1 \rangle \langle y1 \rangle \langle x2 \rangle \langle y2 \rangle \langle x3 \rangle \langle y3 \rangle \ c\)
- 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: \(0x\text{FFFFFF}\)

Set or get the background color as RGB value of the generated image when using `ConvertPathToImage`. The format is \(0x\text{BB} \text{GG} \text{RR}\), where \(\text{BB}\), \(\text{GG}\), and \(\text{RR}\) are hexadecimal numbers in the range \(00\text{FF}\) that represent the values of the blue, green, and red channel respectively. The default \(0x\text{FFFFFF}\) 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

<table>
<thead>
<tr>
<th>Property (get, set):</th>
<th>Single PathImageResolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default:</td>
<td>150</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Property (get, set):</th>
<th>Long SpaceFactor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default:</td>
<td>0.3</td>
</tr>
</tbody>
</table>

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 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). 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 an AlternateArray object 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 the ColorSpace object 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). 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

<details>
<summary>Property (get): Boolean IsMonochrome</summary>

This property is `True` when the image is monochrome.

</details>

### 6.4.13 Mask

<details>
<summary>Property (get): Image Mask</summary>

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.

</details>

### 6.4.14 ObjNumber

<details>
<summary>Property (get): long ObjNumber</summary>

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.

</details>

### 6.4.15 Samples

<details>
<summary>Property (get): Variant Samples</summary>

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.

</details>

### 6.4.16 SMask

<details>
<summary>Property (get, set): Image SMask</summary>

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 $2^{\text{BitsPerComponent}}$ ("white") “fully opaque”. The resolution

</details>
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:

<table>
<thead>
<tr>
<th>Extension</th>
<th>File Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;.bmp&quot;, &quot;.dib&quot;, &quot;.pbm&quot;, &quot;.pgm&quot;, &quot;.pnm&quot;, &quot;.ppm&quot;</td>
<td>Bitmap</td>
</tr>
<tr>
<td>&quot;.jpg&quot;, &quot;.jpe&quot;</td>
<td>JPEG</td>
</tr>
<tr>
<td>&quot;.jb2&quot;</td>
<td>JBIG2</td>
</tr>
<tr>
<td>&quot;.jp2&quot;, &quot;.jpx&quot;</td>
<td>JPEG2000</td>
</tr>
<tr>
<td>&quot;.tif&quot;</td>
<td>TIFF</td>
</tr>
<tr>
<td>&quot;.png&quot;</td>
<td>PNG</td>
</tr>
<tr>
<td>&quot;.gif&quot;, &quot;.jif&quot;</td>
<td>GIF</td>
</tr>
<tr>
<td>&quot;.eps&quot;</td>
<td>EPS</td>
</tr>
</tbody>
</table>

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

**Property (get):** Short Length

This property is superseded by **StringLength**.

6.5.4 QuadPoints

**Property (get, set):** TPDFVector[4] QuadPoints

Get the quad points of the parallelogram that encloses the text. The first point in the array is always located at the lower left corner of the text. The remaining points correspond to the parallelogram points when traveling around the text in counter-clockwise direction. E.g. the vector from the first to the second point always indicates the writing direction of the text. (See also the graphic in the description of **BoundingBox**)

**Note:** This property is not available in the COM interface.

6.5.5 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
6.5.6 Rotation

**Property (get): Single Rotation**

Get the rotation of the string in radians (rad). \((2\pi \text{ rad} = 360 ^\circ)\)

6.5.7 StringLength

**Property (get): Short StringLength**

Get the number of characters in the string.

6.5.8 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). 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 object tells whether a text written with a specific font can be mapped to Unicodes.

6.5.9 TextMatrix

**[Deprecated]Property (get): TransformMatrix TextMatrix**

Get the TransformMatrix object 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.

6.5.10 Width

**Property (get): Single Width**

Get the width of the string in points.
### 6.5.11 XPos, YPos, Position

<table>
<thead>
<tr>
<th>Property (get):</th>
<th>Variant XPos</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property (get):</td>
<td>Variant YPos</td>
</tr>
<tr>
<td>Property (get):</td>
<td>TPDFVector[] Position</td>
</tr>
</tbody>
</table>

**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 the current TransformMatrix. 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:

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

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):

```vbnet
Quad = PDFPARSERLib.GraphicsState.FillColorCMYK
C = Quad And &H7FFFFFFF
M = (Quad \ 65536) And 255
Y = (Quad \ 256) And 255
K = Quad 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:

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

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):

```vbnet
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 the current ColorSpace 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 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.
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 × 256) + M) × 256 + Y) × 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 × 256) + G) × 256 + B.

6.6.29 StrokeColorSpace

**Property (get):** ColorSpace StrokeColorSpace

Get the current ColorSpace 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 `Strings`. 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).

<table>
<thead>
<tr>
<th>Bit Position</th>
<th>Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FixedPitch</td>
<td>All glyphs have the same width.</td>
</tr>
<tr>
<td>2</td>
<td>Serif</td>
<td>Glyphs have serifs.</td>
</tr>
<tr>
<td>3</td>
<td>Symbolic</td>
<td>The font contains characters outside the standard Latin character set.</td>
</tr>
<tr>
<td>4</td>
<td>Script</td>
<td>Glyphs resemble cursive handwriting.</td>
</tr>
<tr>
<td>6</td>
<td>NonSymbolic</td>
<td>Font uses standard Latin character set or a subset of it.</td>
</tr>
<tr>
<td>7</td>
<td>Italic</td>
<td>Glyphs are italic.</td>
</tr>
<tr>
<td>17</td>
<td>AllCap</td>
<td>Font has no lowercase letters.</td>
</tr>
<tr>
<td>18</td>
<td>SmallCap</td>
<td>Lowercase letters are small uppercase letters.</td>
</tr>
<tr>
<td>19</td>
<td>ForceBold</td>
<td>If set, bold glyphs are painted bold even at very small text size.</td>
</tr>
</tbody>
</table>

### 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.

<table>
<thead>
<tr>
<th>Value</th>
<th>PDF Font-File Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>—</td>
<td>This font has no font file.</td>
</tr>
<tr>
<td>1</td>
<td>FontFile</td>
<td>Type 1 font.</td>
</tr>
<tr>
<td>2</td>
<td>FontFile2</td>
<td>TrueType font.</td>
</tr>
<tr>
<td>3</td>
<td>FontFile3</td>
<td>The type of the font is determined by <em>Type</em>.</td>
</tr>
</tbody>
</table>

### 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* 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 the base ColorSpace if 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 space 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

<table>
<thead>
<tr>
<th>Property (get):</th>
<th>Single a</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property (get):</td>
<td>Single b</td>
</tr>
<tr>
<td>Property (get):</td>
<td>Single c</td>
</tr>
<tr>
<td>Property (get):</td>
<td>Single d</td>
</tr>
<tr>
<td>Property (get):</td>
<td>Single e</td>
</tr>
<tr>
<td>Property (get):</td>
<td>Single f</td>
</tr>
</tbody>
</table>

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 \([x \ y \ 1]\) to a vector \([x' \ y' \ 1]\) \(= [x \ y \ 0] \cdot M\), 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 \(\alpha\) counter clockwise is produced by a matrix \([\cos \alpha \sin \alpha \ -\sin \alpha \cos \alpha \ 0 \ 0]\).

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 \(XSkew, YSkew\) and \(-XSkew, YSkew\).
6.9.4 XScaling, YScaling

<table>
<thead>
<tr>
<th>Property (get):</th>
<th>Single XScaling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property (get):</td>
<td>Single YScaling</td>
</tr>
</tbody>
</table>

Get the horizontal and vertical scaling factor.

6.9.5 XSkew, YSkew

<table>
<thead>
<tr>
<th>Property (get):</th>
<th>Single XSkew</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property (get):</td>
<td>Single YSkew</td>
</tr>
</tbody>
</table>

Get the x and y-axis skewing. The transformation matrix \(\begin{bmatrix} 1 & \tan \alpha & \tan \beta & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}\) skews the x-axis by \(\alpha\) and the y-axis by \(\beta\). Skewing sometimes is used to transform a regular font to italic.

6.9.6 XTranslation, YTranslation

<table>
<thead>
<tr>
<th>Property (get):</th>
<th>Single XTranslation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property (get):</td>
<td>Single YTranslation</td>
</tr>
</tbody>
</table>

Get the translation in horizontal and vertical direction. These are the same values as returned by the properties \(a\), \(b\), \(c\), \(d\), \(e\), \(f\) and \(a\), \(b\), \(c\), \(d\), \(e\), \(f\).

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 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 the EmbeddedFile 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 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 and later)
3. Print (PDF 1.2 and later)
4. NoZoom (PDF 1.3 and later)
5. NoRotate (PDF 1.3 and later)
6. NoView (PDF 1.3 and later)
7. ReadOnly (PDF 1.3 and later)
8. Locked (PDF 1.4 and later)
9. ToggleNoView (PDF 1.5 and later)

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 \([x_1, y_1, x_2, y_2]\). Where \((x_1, y_1)\) is the lower left corner of the annotation and \((x_2, y_2)\) 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.

6.12.12TextLabel

**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 QuadPoints

**Property (get):** Variant QuadPoints

Get the quad points of a text markup or a link annotation. The return value is an array of floating-point numbers that represent x- and y-values of the quad points: \(x_1 \ y_1 \ x_2 \ y_2 \ldots \ x_n \ y_n\), where \(n\) is the number of quad points. For
other annotation types, this property is **Nothing**. If the text markup or link annotation has no quad points then the points of the bounding rectangle are returned.

### 6.12.14 URI

**Property (get):** String URI

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

### 6.12.15 Vertices

**Property (get):** Variant Vertices

Get the vertices of a polygon or polyline annotation. The return value is an array of floating-point numbers that represent x- and y-values of the vertices: \( x_1, y_1, x_2, y_2, \ldots, x_n, y_n \) where \( n \) is the number of vertices in the polygon or polyline. For other annotation types or if the polygon/polyline annotation is empty, this property is **Nothing**.

## 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 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:

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 zero 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 PdObjExt.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#

```csharp
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 for the i-th value in the array.

6.16.5 GetEntry

**Method:** PDFObject GetEntry(String Key)

Applies to dictionaries. Return a PDFObject 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 that corresponds 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 **TextStringValue**

**Property (get):** String TextStringValue

Get the correctly decoded value of a PDF string object as a string.

**Note:** In PDF, string objects can be used for any binary data. This property only yields correct strings for PDF string objects that represent human readable text strings. Please consult the PDF Reference 1.7 to determine which string objects can safely be considered as text strings.

6.16.16 **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 DPartRoot Interface

This interface is used to access the document parts information of a PDF 2.0 or a PDF/VT. Please consult the respective standards ISO 32000-2 and ISO 16612-2 for more information on document parts.

6.18.1 NodeNameCount, NodeName

**Property (get):** Integer NodeNameCount

**Method:** String NodeName(Integer index)

Access the document parts' node name list by querying the length of this list with NodeNameCount and getting individual elements with NodeName using the zero based index.

Each name in this list corresponds to a document part node level of the document part hierarchy beginning with the RootNode.

If no node name list is present then NodeNameCount is 0.

6.18.2 RecordLevel

**Property (get):** Integer RecordLevel

Get the zero based level of the document part hierarchy at which each DPartNode corresponds to a component of the document.

6.18.3 RootNode

**Property (get):** DPartNode RootNode

Get a DPartNode interface to the root node in the document parts' hierarchy of nodes.
6.19 DPartNode Interface

6.19.1 ChildrenCount, Child

<table>
<thead>
<tr>
<th>Property (get):</th>
<th>Integer ChildrenCount</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method:</td>
<td>DPartNode Child(Integer index)</td>
</tr>
</tbody>
</table>

Access this node's children by querying their count with `ChildrenCount` and accessing individual elements with `Child` using the zero based `index`. If no children are present then `ChildrenCount` is `0`.

Note that a document part node can either have children or an associated `PageRange`, but not both.

6.19.2 DPM

| Property (get, set): | PDFObject DPM |

Use this property to extract the document part metadata (DPM) associated with this node. For document parts, the metadata takes the form of PDF objects, see `PDFObject`. This property returns the DPM dictionary which may contain as entry values any other PDF objects of type text string, date string, array, dictionary, boolean, integer, or real. This means that for PDF string objects you can safely access the text string by means of the `TextStringValue` property.

6.19.3 PageRange

| Property (get, set): | PageRange PageRange |

Get the page range, i.e. the first and last page number (one-based), associated with this document part node. If this node has no associated page range, then this property evaluates to `Nothing`.

Note that a document part node can either have children (`ChildrenCount, Child`) or an associated page range, but not both.

6.20 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).

6.20.1 TPDFCompliance Enumeration
### TPDFCompliance Table

<table>
<thead>
<tr>
<th>TPDFCompliance</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ePDF11</td>
<td>PDF Version 1.1</td>
</tr>
<tr>
<td>ePDF12</td>
<td>PDF Version 1.2</td>
</tr>
<tr>
<td>ePDF13</td>
<td>PDF version 1.3</td>
</tr>
<tr>
<td>ePDF14</td>
<td>PDF version 1.4 (corresponds to Acrobat 5)</td>
</tr>
<tr>
<td>ePDF15</td>
<td>PDF version 1.5</td>
</tr>
<tr>
<td>ePDF16</td>
<td>PDF version 1.6 (corresponds to Acrobat 7)</td>
</tr>
<tr>
<td>ePDF17</td>
<td>PDF version 1.7, ISO 32000-1</td>
</tr>
<tr>
<td>ePDF18</td>
<td>PDF version 2.0, ISO 32000-2</td>
</tr>
<tr>
<td>ePDF1A</td>
<td>PDF/A 1a, ISO 19005-1, conformance level A</td>
</tr>
<tr>
<td>ePDF1B</td>
<td>PDF/A 1b, ISO 19005-1, conformance level B</td>
</tr>
<tr>
<td>ePDF2A</td>
<td>PDF/A 2a, ISO 19005-2, conformance level A</td>
</tr>
<tr>
<td>ePDF2B</td>
<td>PDF/A 2b, ISO 19005-2, conformance level B</td>
</tr>
<tr>
<td>ePDF2U</td>
<td>PDF/A 2u, ISO 19005-2, conformance level U</td>
</tr>
<tr>
<td>ePDF3A</td>
<td>PDF/A 3a, ISO 19005-3, conformance level A</td>
</tr>
<tr>
<td>ePDF3B</td>
<td>PDF/A 3b, ISO 19005-3, conformance level B</td>
</tr>
<tr>
<td>ePDF3U</td>
<td>PDF/A 3u, ISO 19005-3, conformance level U</td>
</tr>
</tbody>
</table>

Note that only the values listed above are supported.

### 6.20.2 TPDFCompression Enumeration

<table>
<thead>
<tr>
<th>TPDFCompression</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eComprRaw</td>
<td>No compression</td>
</tr>
<tr>
<td>eComprJPEG</td>
<td>Joint Photographic Expert Group</td>
</tr>
<tr>
<td>eComprFlate</td>
<td>Flate compression</td>
</tr>
<tr>
<td>eComprLZW</td>
<td>Lempel-Ziv-Welch</td>
</tr>
<tr>
<td>eComprGroup3</td>
<td>CCITT Fax Group 3</td>
</tr>
<tr>
<td>eComprGroup3_2D</td>
<td>CCITT Fax Group 3 2D</td>
</tr>
</tbody>
</table>
### TPDFCompression Table

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eComprGroup4</td>
<td>CCITT Fax Group 4</td>
</tr>
<tr>
<td>eComprJBIG2</td>
<td>Joint Bi-level Image Experts Group</td>
</tr>
<tr>
<td>eComprJPEG2000</td>
<td>JPEG2000</td>
</tr>
<tr>
<td>eComprUnknown</td>
<td>Unknown compression</td>
</tr>
<tr>
<td>eComprDefault</td>
<td>Apply a default compression which suites the color space of the image.</td>
</tr>
</tbody>
</table>

**Note:** Not all image formats/color depths support all compression types.

### 6.20.3 TPDFContentObject Enumeration

See also method `GetNextObject`.

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

### 6.20.4 TPDFErrorCode Enumeration

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

<table>
<thead>
<tr>
<th>TPDFErrorCode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDF_S_SUCCESS</td>
<td>The operation was completed successfully.</td>
</tr>
<tr>
<td>LIC_E_NOSET, LIC_E_NOTFOUND, ...</td>
<td>Various license management related errors.</td>
</tr>
<tr>
<td>PDF_E_FILEOPEN</td>
<td>Failed to open the file.</td>
</tr>
<tr>
<td>PDF_E_FILECREATE</td>
<td>Failed to create the file.</td>
</tr>
<tr>
<td>PDF_E_PASSWORD</td>
<td>The authentication failed due to a wrong password.</td>
</tr>
<tr>
<td>PDF_E_UNKSECHANDLER</td>
<td>The file uses a proprietary security handler, e.g. for a proprietary digital rights management (DRM) system.</td>
</tr>
<tr>
<td>PDF_E_XFANEEDSRENDERING</td>
<td>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.</td>
</tr>
</tbody>
</table>

### 6.20.5 TPDFObjectType Enumeration

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eTypeNull</td>
<td>The PDF null object.</td>
</tr>
<tr>
<td>eTypeBoolean</td>
<td>A PDF object that represents a Boolean value (true or false).</td>
</tr>
<tr>
<td>eTypeInteger</td>
<td>A PDF numeric object that represents an integer number.</td>
</tr>
<tr>
<td>eTypeReal</td>
<td>A PDF numeric object that represents a real-valued number.</td>
</tr>
<tr>
<td>eTypeString</td>
<td>A PDF string object.</td>
</tr>
<tr>
<td>eTypeName</td>
<td>A PDF name object.</td>
</tr>
<tr>
<td>eTypeArray</td>
<td>A PDF array object.</td>
</tr>
<tr>
<td>eTypeDictionary</td>
<td>A PDF dictionary object.</td>
</tr>
</tbody>
</table>
6.20.6 TPDFOrientation Enumeration

**TPDFOrientation Table**

<table>
<thead>
<tr>
<th>TPDFOrientation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eOrientationUndef</td>
<td>Undefined</td>
</tr>
<tr>
<td>eOrientationTopLeft</td>
<td>Pages appear in columns, from bottom to top and right to left relative to page orientation.</td>
</tr>
<tr>
<td>eOrientationTopRight</td>
<td>Pages appear in columns, from bottom to top and left to right relative to page orientation.</td>
</tr>
<tr>
<td>eOrientationBottomRight</td>
<td>Pages appear in columns, from top to bottom and left to right relative to page orientation.</td>
</tr>
<tr>
<td>eOrientationBottomLeft</td>
<td>Pages appear in columns, from top to bottom and right to left relative to page orientation.</td>
</tr>
<tr>
<td>eOrientationLeftTop</td>
<td>Pages appear in rows, from right to left and bottom to top relative to page orientation.</td>
</tr>
<tr>
<td>eOrientationRightTop</td>
<td>Pages appear in rows, from left to right and bottom to top relative to page orientation.</td>
</tr>
<tr>
<td>eOrientationRightBottom</td>
<td>Pages appear in rows, from left to right and top to bottom relative to page orientation.</td>
</tr>
<tr>
<td>eOrientationLeftBottom</td>
<td>Pages appear in rows, from right to left and top to bottom relative to page orientation.</td>
</tr>
</tbody>
</table>

6.20.7 TPDFTextExtractConfiguration Enumeration

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eTECBreakTextState</td>
<td>Start new text object, if text state changes (font, font size, horizontal scaling). Set this flag, if text state is important to you.</td>
</tr>
<tr>
<td>eTECBreakGraphicsState</td>
<td>Start new text object, if graphics state changes (color). Set this flag, if the color is important to you.</td>
</tr>
</tbody>
</table>
| **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” |
| **eTECPoS MergeSingleSpace** | 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. |
| **eTECPoS MergeMultiSpace** | 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 6

- [Java] Changed minimal supported Java language version to 7 [previously 6].
- [PHP] Removed all versions of the PHP interface.
- [.NET] New availability of this product as NuGet package for Windows, macOS and Linux.
- [.NET] New support for .NET Core versions 1.0 and higher. The support is restricted to a subset of the operating systems supported by .NET Core, see Operating Systems.
- [.NET] Changed platform support for NuGet packages: The platform "AnyCPU" is now supported for .NET Framework projects.

7.2 Changes in Version 5

- [PHP] New extension PHP 7.3 (non thread safe) for Linux.

7.3 Changes in Version 4.12

- New HTTP proxy setting in the GUI license manager.
- Improved performance when extracting resources for certain documents.

7.4 Changes in Version 4.11

- New support for reading PDF 2.0 documents.
- Improved repair of corrupt image streams.
- [PHP] New Interface for Windows and Linux. Supported versions are PHP 5.6 & 7.0 (non thread safe). The PdfParser PHP interface is contained in the 3-Heights™ PDF Tools PHP5.6 Extension and the 3-Heights™ PDF Tools PHP7.0 Extension.
- [C] Changed 32-bit binaries on Windows that link to the API need to be recompiled due to a change of the used mangling scheme.
- [.NET, C, COM, Java, PHP] New Interface DPartNode, Interface DPartRoot, and Interface PageRange to allow the extraction of document parts from a PDF 2.0 or a PDF/VT document.

Interface Annotation


Interface Document

Interface DPartNode

- [.NET, C, COM, Java, PHP] New property DPM to extract a node's document part metadata.
- [.NET, C, COM, Java, PHP] New property PageRange to extract a node's page range.

Interface DPartRoot

- [.NET, C, COM, Java, PHP] New property NodeNameCount and method NodeName for extracting the document parts' node name list.
- [.NET, C, COM, Java, PHP] New property RecordLevel for extracting the document parts' record level.
- [.NET, C, COM, Java, PHP] New property RootNode to access the root node in the document part hierarchy.

Interface PageRange

- [.NET, C, COM, Java, PHP] New properties First and Last to access the first and last page number in a page range.

Interface PDFObject

- [.NET, C, COM, Java, PHP] New property TextStringValue to extract the value of a string PDF object as a decoded string.

Interface Text

- [.NET, C, Java, PHP] New property QuadPoints for extracting the enclosing parallelogram.

7.5 Changes in Version 4.10

- Improved robustness against corrupt input PDF documents.
- [C] Clarified Error handling of TPdfStreamDescriptor functions.
- New support of overlapping code ranges in font's ToUnicode tables (used for text extraction).

7.6 Changes in Version 4.9

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

7 This has no effect on neither the .NET, Java, nor COM API

Interface Document

- [.NET, C, Java] New method OpenStream().
Interface Content

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

### 7.7 Changes in Version 4.8

- **New** support for ToUnicode mappings that map characters to sequences of Unicodes (longer than 1). This includes special ligatures and surrogate pairs.
- **Improved** space width heuristic. 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.
- **Improved** creation of annotation appearances to use less memory and processing time.
- **Added** repair functionality for TrueType font programs whose glyphs are not ordered correctly.

Interface Font


Interface Document

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

**Contact**
PDF Tools AG
Brown-Boveri-Strasse 5
8050 Zürich
Switzerland
pdfsales@pdf-tools.com