User Manual

3-Heights™ PDF Merge Split API

PDF/A compliant

Version 4.10

PDF-TOOLS.COM
Premium PDF Technology

Contents

1 Introduction 4
1.1 Features 4
1.2 Interfaces 5
1.3 Operating Systems 5

2 Installation and Deployment 6
2.1 Windows 6
2.2 Unix 6
2.2.1 All Unix Platforms 7
2.2.2 macOS 8
2.3 Interfaces 8
2.3.1 Development 8
2.3.2 Deployment 10
2.4 Interface Specific Installation Steps 11
2.4.1 COM Interface 11
2.4.2 Java Interface 11
2.4.3 .NET Interface 12
2.4.4 C Interface 12
2.4.5 PHP Interface 12
2.5 Uninstall, Install a New Version 13
2.6 Note about the Evaluation License 13

3 License Management 14
3.1 License Installation and Management 14
3.1.1 Graphical License Manager Tool 14
List all installed license keys 14
Add and delete license keys 14
Display the properties of a license 14
3.1.2 Command Line License Manager Tool 14
3.2 License Selection and Precedence 15
3.2.1 Selection 15
3.2.2 Precedence 16
3.3 Key Update 16
3.4 License activation 16
3.4.1 Activation 16
3.4.2 Reactivation 17
3.4.3 Deactivation 17
3.5 Offline Usage 17
3.5.1 First Step: Create a Request File 18
3.5.2 Second Step: Use Form on Website 18
3.5.3 Third Step: Apply the Response File 18
3.6 License Key Versions 19
3.7 License Key Storage 19
3.7.1 Windows 19
3.7.2 macOS 19
3.7.3 Unix/Linux 19
3.8 Troubleshooting 20
3.8.1 License key cannot be installed 20
3.8.2 License is not visible in license manager 20

3.8.3 License is not found at runtime ... 20
3.8.4 Eval watermark is displayed where it should not ... 20

4 Programming Interfaces ... 22
4.1 Visual Basic 6 ... 22
4.2 .NET ... 23
4.2.1 Visual Basic ... 23
4.2.2 C# ... 24
4.2.3 Deployment ... 25
4.2.4 Troubleshooting: TypeInitializationException ... 25
Troubleshooting: DllNotFoundException ... 25
Troubleshooting: BadImageFormatException ... 26

5 User’s Guide ... 27
5.1 Basics ... 27
5.2 How to Create PDF/A Compliant Documents ... 27
5.3 How to Efficiently Use the API ... 28
5.3.1 How to Copy Pages ... 28
5.3.2 Operation on Multiple Documents ... 29
5.3.3 Features and their Impact on Performance ... 29
5.4 Error Handling ... 29

6 Programmer’s Reference ... 31
6.1 `InDoc` Interface ... 31
6.1.1 `Close` ... 31
6.1.2 `CropBox` ... 31
6.1.3 `ErrorCode` ... 31
6.1.4 `ErrorMessage` ... 32
6.1.5 `GetInfoEntry` ... 32
6.1.6 `GetXMPMetadata` ... 32
6.1.7 `GetXMPMetadataMem` ... 32
6.1.8 `MediaBox` ... 33
6.1.9 `Open` ... 33
6.1.10 `OpenMem` ... 33
6.1.11 `Page` ... 34
6.1.12 `PageCount` ... 34
6.1.13 `Rotate` ... 34
6.2 `OutDoc` Interface ... 34
6.2.1 `AddAssociatedFile` ... 34
6.2.2 `AddEmbeddedFile` ... 35
6.2.3 `AddOutlineItem` ... 35
6.2.4 `AddOutlineItem2` ... 37
6.2.5 `Author` ... 37
6.2.6 `Close` ... 37
6.2.7 `CopyAssociatedFiles` ... 38
6.2.8 `CopyAttributes` ... 38
6.2.9 `CopyEmbeddedFiles` ... 38
6.2.10 `CopyForms` ... 38
6.2.11 `CopyLogicalStructure` ... 39
6.2.12 `CopyMetadata` ... 39
6.2.13 `CopyOptionalContent` ... 39
6.2.14 `CopyOutlines` ... 39
6.2.15 `CopyOutlineItems` ... 39

6.2.16 CopyOutlineItems2 39
6.2.17 CopyOutputIntent 40
6.2.18 CopyPages 41
6.2.19 CopyPages2 41
6.2.20 CopyViewerProperties 41
6.2.21 Create 41
6.2.22 CreateInMemory 43
6.2.23 ErrorCode 43
6.2.24 ErrorMessage 43
6.2.25 FlattenAnnotations 43
6.2.26 FlattenFormFields 43
6.2.27 FlattenSigAppearance 44
6.2.28 GetPdf 44
6.2.29 InfoEntry 44
6.2.30 Keywords 45
6.2.31 LicenseIsValid 45
6.2.32 Linearize 45
6.2.33 MergeOptionalContent 45
6.2.34 OptimizeResources 45
6.2.35 OutputIntent 46
6.2.36 PageLayout 46
6.2.37 PageMode 46
6.2.38 ProductVersion 46
6.2.39 RemoveNamedDests 46
6.2.40 Rotate 47
6.2.41 SetLicenseKey 47
6.2.42 SetOpenAction 47
6.2.43 SetViewerPreference 47
6.2.44 SetXMPMetadata 48
6.2.45 SetXMPMetadataMem 48
6.2.46 Subject 48
6.2.47 Title 48
6.3 Enumerations 48
6.3.1 TPdfCopyOption 49
6.3.2 TPDFDestMode 52
6.3.3 TPDFErrorCode 53
6.3.4 TPDFPermission 54
6.3.5 TPDFPageLayout 55
6.3.6 TPDFPageMode 55

7 Examples 57

8 Version History 58
8.1 Changes in Version 4.10 58
8.2 Changes in Version 4.9 58
8.3 Changes in Version 4.8 58

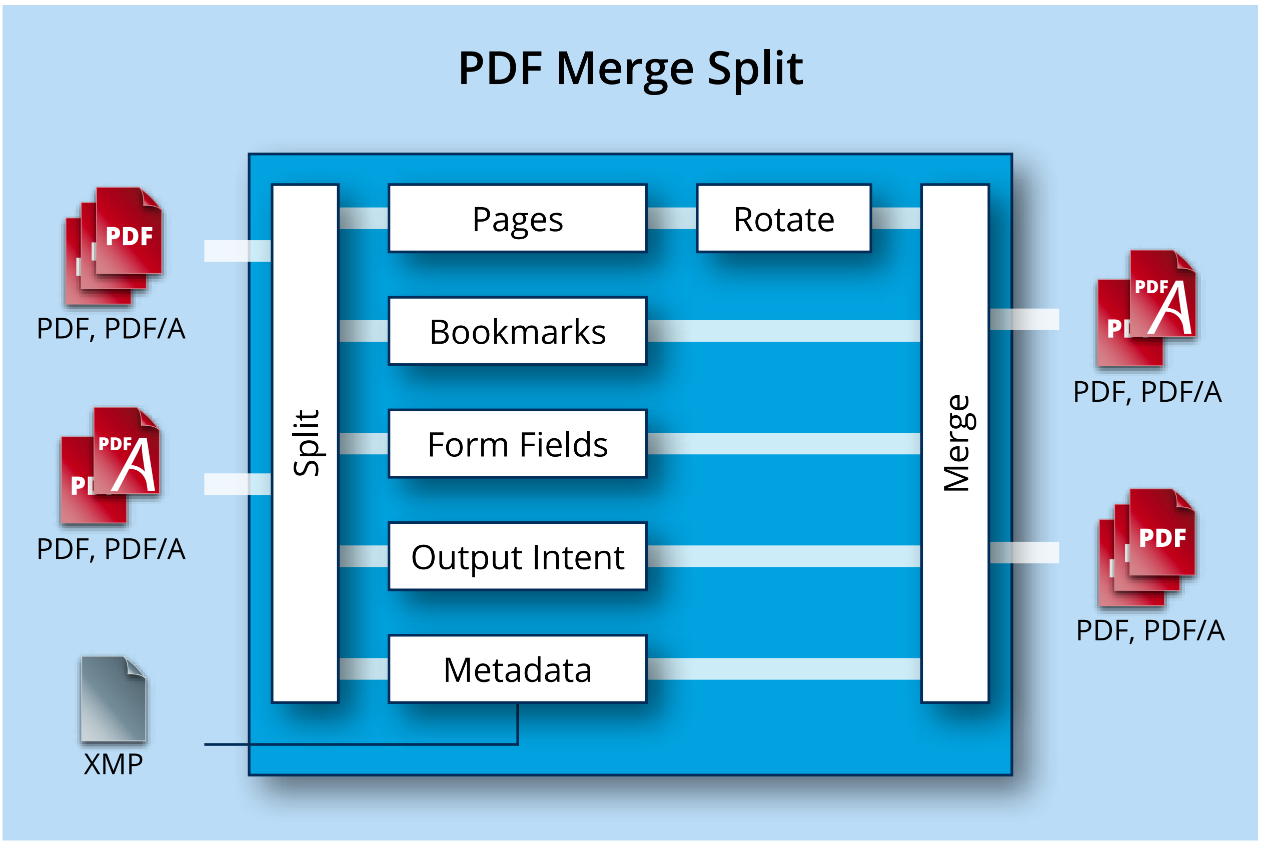
9 Licensing, Copyright, and Contact 59

1 Introduction

The 3-Heights™ PDF Merge Split API is a component for splitting and merging the pages of PDF documents with useful additional functions.

In addition to its main functions of splitting and merging, the 3-Heights™ PDF Merge Split API can also rotate pages, copy or add metadata and other document attributes such as document outlines (bookmark), form fields, color profiles for output devices and much more, as well as flattening form fields.

The 3-Heights™ PDF Merge Split API can operate on multiple input and output documents in one processing step. A special feature is the component's ability to process and create PDF/A-compliant files.



1.1 Features

The 3-Heights™ PDF Merge Split API comes with the following features:

- Merge different PDF documents or pages thereof to form a single PDF document
- Split a PDF document of many pages into a number of smaller PDF documents
- Process PDF/A documents: If all the input documents are PDF/A, then the output is PDF/A.
- Rotate pages
- Flatten or remove form fields and annotations
- Set or copy the color profile for the output device (output intent)
- Set or copy document information and metadata (XMP)
- Extract the number of pages, the media box and crop box of a PDF document
- Extract XMP metadata from a PDF document
- Add embedded files to a PDF document
- Optimize page resources when merging PDF documents
- Set passwords and permission flags

- Process from the file system and from memory
- Copy or remove outlines (bookmarks) and create custom outlines
- Merge or remove document structure information
- Remove named destinations
- Set document information entries (title, author, . . .)
- Write a linearized PDF (fast web view)
- Set the page mode, initial page layout, and open action
- Split vertical or horizontal double pages into single pages

1.2 Interfaces

The following interfaces are available

- C
- .NET
- Java
- COM
- PHP

1.3 Operating Systems

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

- Windows 7, 8, 8.1, 10 – 32 and 64 bit
- Windows Server 2008, 2008 R2, 2012, 2012 R2, 2016 – 32 and 64 bit
- HP-UX 11i and later PA-RISC2.0 – 32 bit
- HP-UX 11i and later ia64 (Itanium) – 64 bit
- IBM AIX 6.1 and later – 64 bit
- Linux 2.6 – 32 and 64 bit
- Oracle Solaris 2.8 and later, SPARC and Intel
- FreeBSD 4.7 and later (32 bit) or FreeBSD 9.3 and later (64 bit, on request)
- macOS 10.4 and later – 32 and 64 bit

2 Installation and Deployment

2.1 Windows

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

The installation of the software requires the following steps.

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

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

4. (Optional) Register your license key using the License Management.
5. Identify which interface you are using. Perform the specific installation steps for that interface described in chapter Interface Specific Installation Steps

2.2 Unix

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

The Unix version of the 3-Heights™ PDF Merge Split API provides three interfaces:

- Java interface
- Native C interface
- PHP interface - Linux only

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

File Description

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

2.2.1 All Unix Platforms

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

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

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

   ```
   ldd libPdfSplMrgAPI.so
   ```

   - *On AIX*:

   ```
   dump -H libPdfSplMrgAPI.so
   ```

   In case the above reports any missing libraries you have two options:

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

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 libPdfSplMrgAPI.dylib libPdfSplMrgAPI.jnilib
```

2.3 Interfaces

The 3-Heights™ PDF Merge Split API provides five different interfaces. The installation and deployment of the software depend on the interface you are using. The table below shows the supported interfaces and examples with which programming languages they can be used.

Interface	Programming Languages
`.NET`	The MS software platform .NET can be used with any .NET capable programming language such as: ■ C# ■ VB .NET ■ J# ■ others This interface is available in the Windows version only.
`Java`	The Java interface is available on all platforms.
`COM`	The component object model (COM) interface can be used with any COM-capable programming language, such as: ■ MS Visual Basic ■ MS Office Products such as Access or Excel (VBA) ■ C++ ■ VBScript ■ others This interface is available in the Windows version only.
`C`	The native C interface is for use with C and C++. This interface is available on all platforms.
`PHP`	The PHP interface is available on Windows and Linux. Supported PHP versions are PHP 5.6 & 7.0 (Non Thread Safe).

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 five different interfaces is shown in table Files for Development. The files are split in four categories:

Req. This file is required for this interface.

Opt. This file is optional. See also table File Description to identify which files are required for your application.

Doc. This file is for documentation only.

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

Files for Development

Name	.NET	Java	COM	C	PHP
`bin\‹platform›\PdfSplMrgAPI.dll`	Req.	Req.	Req.	Req.	Req.
`bin\*NET.dll`	Req.				
`bin\*NET.xml`	Doc.				
`bin\‹platform›\php*_pdftools.dll`					Req.
`doc\*.pdf`	Doc.	Doc.	Doc.	Doc.	Doc.
`doc\PdfSplMrgAPI.idl`			Doc.		
`doc\javadoc\*.*`		Doc.			
`doc\pdfsplmrg_doc.php` `and pdftoolsenums_doc.php`					Doc.
`include\pdfsplmrgapi_c.h`				Req.	
`include\*.*`				Opt.	
`jar\MSPA.jar`		Req.			
`lib\‹platform›\PdfSplMrgAPI.lib`				Req.	
`samples\*.*`	Doc.	Doc.	Doc.	Doc.	Doc.

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

File Description

Name	Description
`bin\‹platform›\PdfSplMrgAPI.dll`	This is the DLL that contains the main functionality (required).
`bin\*NET.dll`	The .NET assemblies are required when using the .NET interface. The files `bin\*NET.xml` contain the corresponding XML documentation for MS Visual Studio.
`doc\*.*`	Various documentations.
`include\*.*`	Contains files to include in your C / C++ project.
`lib\‹platform›\PdfSplMrgAPI.lib`	The object file library needs to be linked to the C/C++ project.
`jar\MSPA.jar`	The Java API archive.
`bin\‹platform›\php*_pdftools.dll`	The `PdfTools` PHP extension must be added to the PHP extension directory.

File	Description
samples*.*	Contains sample programs in different programming languages.

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 five different interfaces is shown in the table below.

Files for Deployment

Name	.NET	Java	COM	C	PHP
bin\‹platform›\PdfSplMrgAPI.dll	Req.	Req.	Req.	Req.	Req.
bin*NET.dll	Req.				
jar\MSPA.jar		Req.			
bin\‹platform›\php*_pdftools.dll					Req.

The deployment of an application works as described below:

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

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

1. The developed and compiled application consists of the file `application.exe`. Color profiles are not used.
2. The application uses the COM interface and is distributed on Windows only.
   - The main DLL `PdfSplMrgAPI.dll` must be distributed.
3. All files are copied to the target location using a batch script. This script contains the following commands:

```
copy application.exe %targetlocation%\.
copy PdfSplMrgAPI.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%\PdfSplMrgAPI.dll.
```

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

2.4 Interface Specific Installation Steps

2.4.1 COM Interface

Registration Before you can use the 3-Heights™ PDF Merge Split 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 PdfSplMrgAPI.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›\PdfSplMrgAPI.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 Merge Split API, you need to use the `regsvr32` from the directory `%SystemRoot%\SysWOW64` instead of `%System-Root%\System32`.[2]

If the registration process succeeds, a corresponding dialog window is displayed. The registration can also be done silently (e.g. for deployment) using the switch `/s`.

Other Files The other DLLs do not need to be registered, but for simplicity it is suggested that they reside in the same directory as the `PdfSplMrgAPI.dll`.

2.4.2 Java Interface

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

For compilation and execution When using the Java interface, the Java wrapper `jar\MSPA.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\MSPA.jar" sample.java
```

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

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

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

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

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

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

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

2.4.3 .NET Interface

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

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

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

The native DLL `PdfSplMrgAPI.dll` is called by the .NET assembly `PdfSplMrgNET.dll`.

`PdfSplMrgAPI.dll` must be found at execution time by the Windows operating system. The common way to do this is adding `PdfSplMrgAPI.dll` as an existing item to the project and set its property "Copy to output directory" to "Copy if newer".

Alternatively the directory where `PdfSplMrgAPI.dll` resides can be added to the environment variable `%Path%` or it can simply be copied manually to the output directory.

2.4.4 C Interface

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

2.4.5 PHP Interface

Note: The descriptions below are valid for Unix-type systems. On a Windows system the libraries have a file extension `dll` instead of `so`.

The PHP interfaces for all 3-Heights™ products are contained in a sole "`PdfTools`" PHP extension. If multiple 3-Heights™ products are used, this extension is needed only once. Supported PHP versions are PHP 5.6 and 7.0 (both non thread-safe). The corresponding PHP extension libraries are `php56_pdftools.so` and `php70_pdftools.so`, henceforth summarized as `php‹xy›_pdftools.so`.

General Steps

- Copy the library `php‹xy›_pdftools.so` to the PHP extensions directory. This directory is configured in the PHP section of your PHP configuration file `php.ini` in the key `extension_dir`.
- Add the following line to the PHP section of your PHP configuration file `php.ini`:

  ```
  extension=php‹xy›_pdftools.so
  ```

- Make sure that the native library `libPdfSplMrgAPI.so` is located in one of the system's library directories[3].

Additional Remarks for Command Line Use of PHP

- You can print out the current PHP configuration as configured in `php.ini` on the command line with:

  ```
  php -i
  ```

- You can check whether PHP loads the `PdfTools` extension with:

  ```
  php -m
  ```

Additional Remarks for Use in a Web Server

- You can locate the currently active PHP configuration file `php.ini` and report loaded modules with the following test PHP script:

  ```
  <?php
  phpinfo();
  ?>
  ```

- For an Apache web server to successfully locate and load the native library `libPdfSplMrgAPI.so` follow these steps:
  a. Create a file `/etc/ld.so.conf.d/pdf-tools.conf` that contains the full path to the directory where `libPdfSplMrgAPI.so` resides.
  b. Execute the command:

     ```
     sudo ldconfig
     ```

  c. Restart the web server.
  d. Reload the test PHP script and check whether there is an entry for the `PdfTools` extension.

2.5 Uninstall, Install a New Version

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

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

2.6 Note about the Evaluation License

With the evaluation license the 3-Heights™ PDF Merge Split API automatically adds a watermark to the output files.

3 License Management

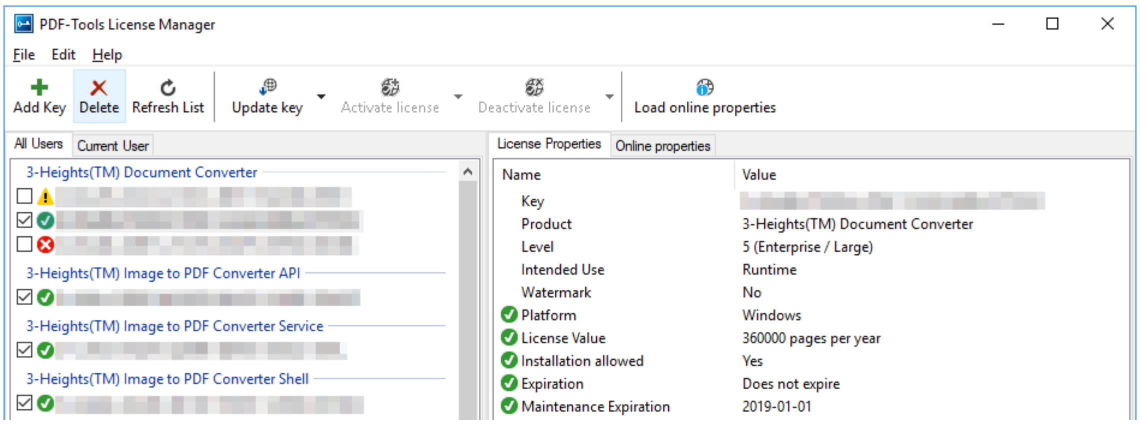
3.1 License Installation and Management

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

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

3.1.1 Graphical License Manager Tool

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



3-Heights(TM) Document Converter
- ☐ ⚠ [redacted]
- ☑ ✓ [redacted]
- ☐ ✗ [redacted]

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

3-Heights(TM) Image to PDF Converter Service
- ☑ ✓ [redacted]

3-Heights(TM) Image to PDF Converter Shell
- ☑ ✓ [redacted]

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

List all installed license keys

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

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

Add and delete license keys

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

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

Display the properties of a license

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

3.1.2 Command Line License Manager Tool

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

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

```
licmgr
```

List all installed license keys:

```
licmgr list
```

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

Add and delete license keys:

Install new license key:

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

Delete old license key:

```
licmgr delete 1-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

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

g	For all users
u	Current user

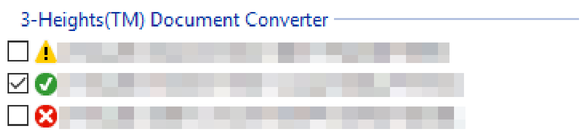
3.2 License Selection and Precedence

3.2.1 Selection

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

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

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



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

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

3.2.2 Precedence

License keys are considered in the following order:

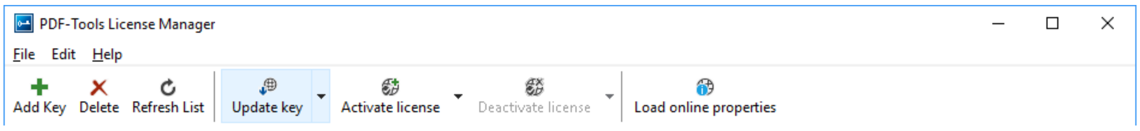
1. License key passed at runtime.
2. License selected for the current user
3. License selected for the current user (legacy key format)
4. License selected for all users
5. License selected for all users (legacy key format)

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

3.3 Key Update

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

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



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

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

3.4 License activation

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

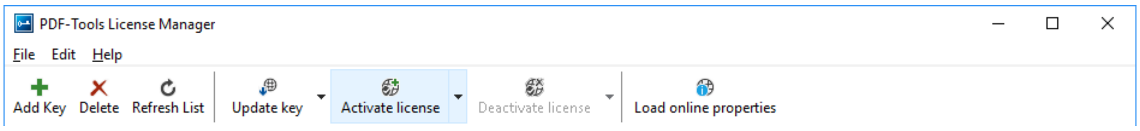
The license activation is tied to a specific computer. If the license is installed at user scope, the activation is also tied to that specific user. The same license key can be activated multiple times, if the license quantity is larger than 1.

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

3.4.1 Activation

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

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



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

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

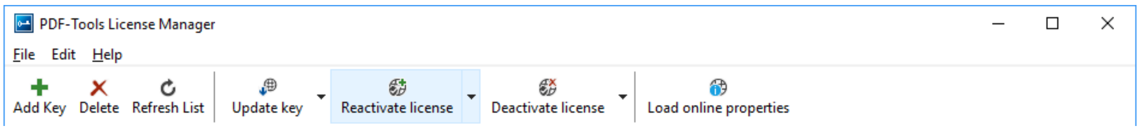
Note that the key has to be installed first.

3.4.2 Reactivation

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

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

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



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

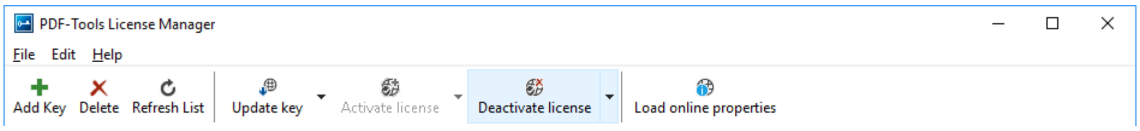
```
licmgr reactivate 1-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

3.4.3 Deactivation

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

The process for deactivation is similar to the activation process.

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



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

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

3.5 Offline Usage

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

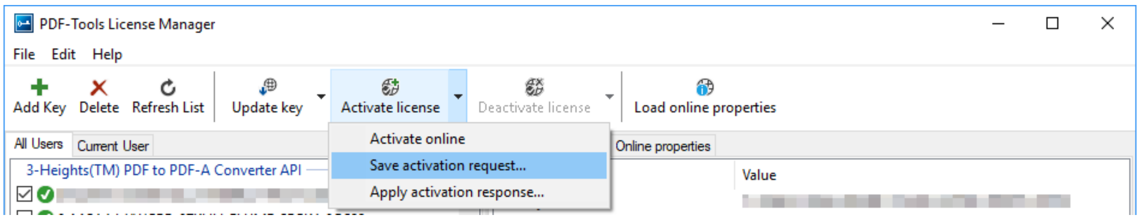
- License Activation
- License Reactivation

- License Deactivation
- Key Update

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

3.5.1 First Step: Create a Request File

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



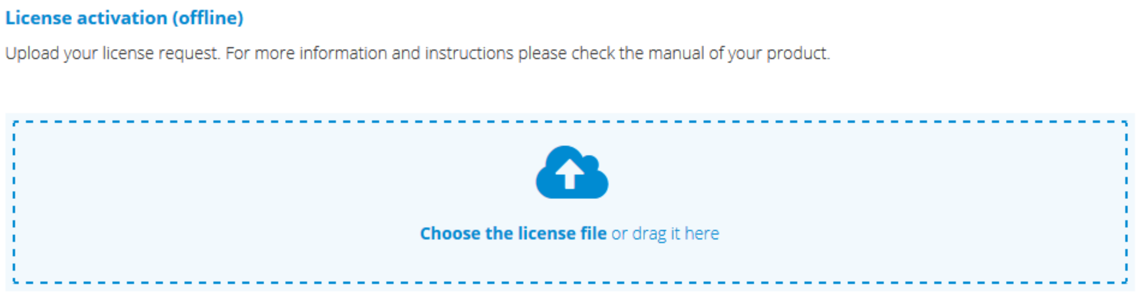
With the Command Line Interface use the `-fs` option to specify the destination path of the request file:

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

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

3.5.2 Second Step: Use Form on Website

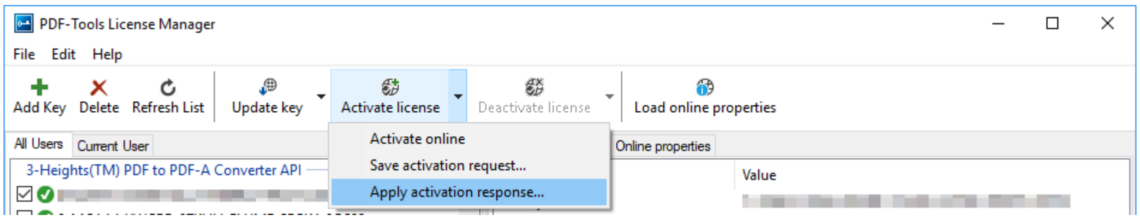
Open the following website in a web browser: `http://www.pdf-tools.com/pdf20/en/mypdftools/licenses-kits/license-activation/` Upload the request by dragging it onto the marked area:


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

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

3.5.3 Third Step: Apply the Response File

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



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

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

3.6 License Key Versions

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

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

3.7 License Key Storage

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

3.7.1 Windows

The license keys are stored in the registry:

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

3.7.2 macOS

The license keys are stored in the file system:

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

3.7.3 Unix/Linux

The license keys are stored in the file system:

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

Note: The user, group and permissions of those directories are set solely by the license manager tool. It may be necessary to change permissions to make the licenses readable for all users. Example:

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

3.8 Troubleshooting

3.8.1 License key cannot be installed

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

Possible causes:

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

Solution

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

3.8.2 License is not visible in license manager

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

Possible causes:

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

Solution

Use a current version of the license manager application.

3.8.3 License is not found at runtime

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

Possible causes:

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

Solution

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

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

3.8.4 Eval watermark is displayed where it should not

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

Possible causes:

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

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

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

Solution

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

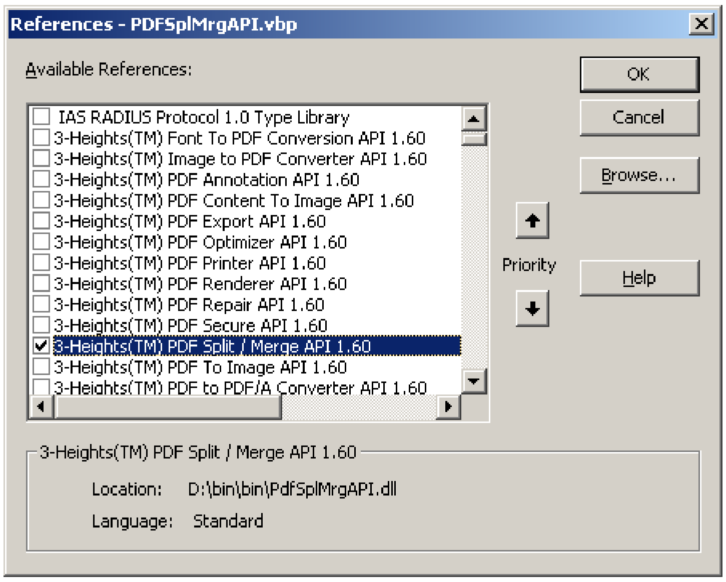
4 Programming Interfaces

4.1 Visual Basic 6

After installing the 3-Heights™ PDF Merge Split 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 Merge Split API component to your project.


References - PDFSplMrgAPI.vbp

Available References:

- [ ] IAS RADIUS Protocol 1.0 Type Library
- [ ] 3-Heights(TM) Font To PDF Conversion API 1.60
- [ ] 3-Heights(TM) Image to PDF Converter API 1.60
- [ ] 3-Heights(TM) PDF Annotation API 1.60
- [ ] 3-Heights(TM) PDF Content To Image API 1.60
- [ ] 3-Heights(TM) PDF Export API 1.60
- [ ] 3-Heights(TM) PDF Optimizer API 1.60
- [ ] 3-Heights(TM) PDF Printer API 1.60
- [ ] 3-Heights(TM) PDF Renderer API 1.60
- [ ] 3-Heights(TM) PDF Repair API 1.60
- [ ] 3-Heights(TM) PDF Secure API 1.60
- [x] 3-Heights(TM) PDF Split / Merge API 1.60
- [ ] 3-Heights(TM) PDF To Image API 1.60
- [ ] 3-Heights(TM) PDF to PDF/A Converter API 1.60

3-Heights(TM) PDF Split / Merge API 1.60

Location: D:\bin\bin\PdfSplMrgAPI.dll

Language: Standard

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

```
Private Sub Command1_Click()
  Dim InDoc As New PDFSPLMRGAPILib.InDoc
  Dim OutDoc1 As New PDFSPLMRGAPILib.OutDoc
  Dim OutDoc2 As New PDFSPLMRGAPILib.OutDoc

  ' Setup what to copy
  Dim CopyOptions = ePdfCopyAnnotations Or _
    ePdfCopyFormFields Or _
    ePdfCopyLinks Or _
    ePdfCopyLogicalStructure Or _
    ePdfCopyNamedDestinations Or _
    ePdfCopyOutlines Or _
    ePdfCopyAssociatedFiles Or _
    ePdfMergeOCGs

  'Create 2 output files, one of them being encrypted, printing allowed
  OutDoc1.Create "P:\PDF\out1.pdf", "", "owner", ePermPrint
  OutDoc2.Create "P:\PDF\out2.pdf"
  InDoc.Open "P:\PDF\input.pdf"
  OutDoc1.CopyAttributes InDoc
```

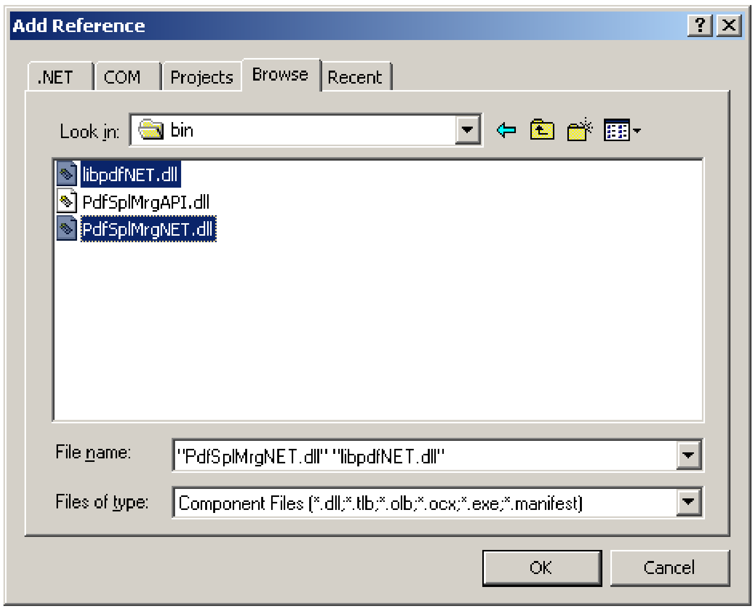
```
    OutDoc1.CopyPages2 InDoc, 1, 2, CopyOptions
    OutDoc2.CopyAttributes InDoc
    OutDoc2.CopyPages2 InDoc, 3, -1, CopyOptions

    InDoc.Close
    OutDoc1.Close
    OutDoc2.Close
End Sub
```

4.2 .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 Merge Split 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` and `PdfSplMrgNET.dll`.
   Add them to the project as shown below:

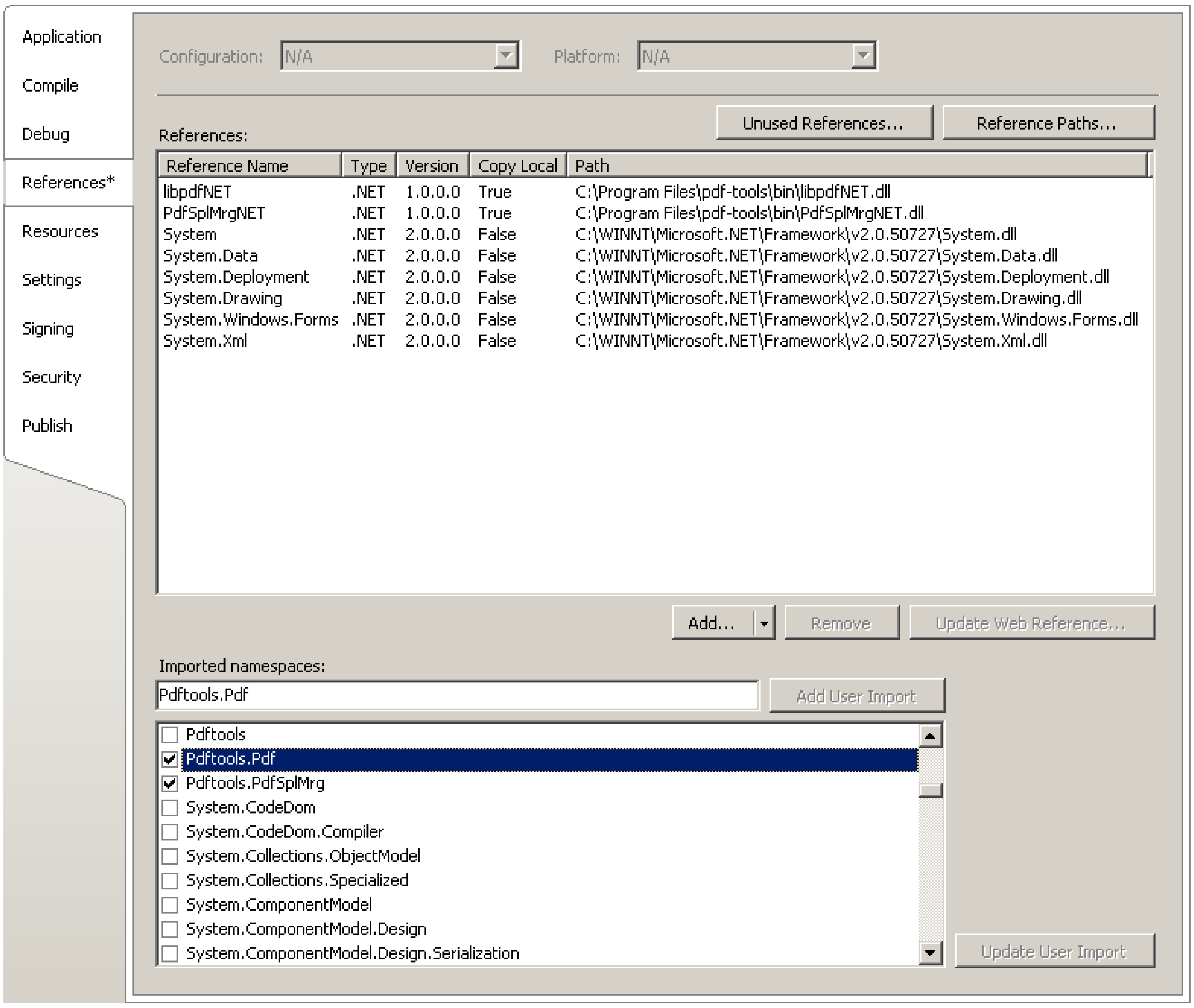


3. Import namespaces (Note: This step is optional, but useful.)
4. Write your code.

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

4.2.1 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`, and `Pdftools.PdfSpMrg`.
   You should now have settings similar as in the screenshot below:



Configuration: N/A Platform: N/A

References:

Reference Name	Type	Version	Copy Local	Path
libpdfNET	.NET	1.0.0.0	True	C:\Program Files\pdf-tools\bin\libpdfNET.dll
PdfSplMrgNET	.NET	1.0.0.0	True	C:\Program Files\pdf-tools\bin\PdfSplMrgNET.dll
System	.NET	2.0.0.0	False	C:\WINNT\Microsoft.NET\Framework\v2.0.50727\System.dll
System.Data	.NET	2.0.0.0	False	C:\WINNT\Microsoft.NET\Framework\v2.0.50727\System.Data.dll
System.Deployment	.NET	2.0.0.0	False	C:\WINNT\Microsoft.NET\Framework\v2.0.50727\System.Deployment.dll
System.Drawing	.NET	2.0.0.0	False	C:\WINNT\Microsoft.NET\Framework\v2.0.50727\System.Drawing.dll
System.Windows.Forms	.NET	2.0.0.0	False	C:\WINNT\Microsoft.NET\Framework\v2.0.50727\System.Windows.Forms.dll
System.Xml	.NET	2.0.0.0	False	C:\WINNT\Microsoft.NET\Framework\v2.0.50727\System.Xml.dll

Imported namespaces:

Pdftools.Pdf

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

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

Example:

```
Dim indoc As New Pdftools.PdfSplMrg.InDoc
Dim outdoc As New Pdftools.PdfSplMrg.OutDoc
indoc.Open(...)
...
```

4.2.2 C#

3. Add the following namespaces:

Example:

```
using Pdftools.Pdf;
using Pdftools.PdfSpMrg;
```

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

Example:

```
using (InDoc indoc = new InDoc())
{
    indoc.Open(...);
    using (OutDoc outdoc = new OutDoc())
    {
        ...
    }
}
```

4.2.3 Deployment

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

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

4.2.4 Troubleshooting: TypeInitializationException

The most common issue when using the .NET interface is that the correct native DLL `PdfSplMrgAPI.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 `PdfSplMrgAPI.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 `PdfSplMrgAPI.dll` could not be found at execution time.

Resolve this by either:

- adding `PdfSplMrgAPI.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 `PdfSplMrgAPI.dll` resides to the environment variable `%Path%`, or
- copying `PdfSplMrgAPI.dll` to the output directory of your project.

Troubleshooting: BadImageFormatException

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

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

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

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

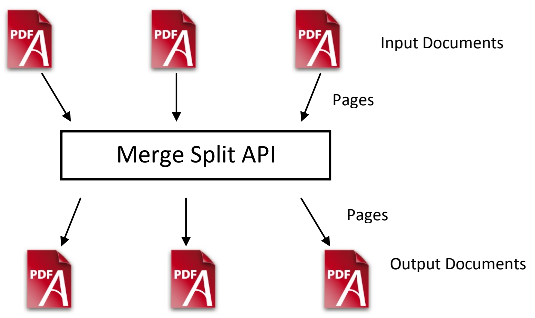
x64 This means, that the application will always run as 64-bit process. As a consequence the application will not run on a 32-bit Windows system.

5 User's Guide

5.1 Basics

The 3-Heights™ PDF Merge Split API uses a multiple-in/multiple-out architecture. This means it can keep multiple inputs and outputs open and copy pages from the input to the output documents.

This allows for efficient merge and split operations.



Input and output documents can be files or streams.

The 3-Heights™ PDF Merge Split API not only merges and splits pages, but also resources (images, fonts, color spaces, etc.), form fields and outlines (bookmarks). This means if a large document is split into several smaller documents, and they are then merged back into one document, it should result in the original document.

5.2 How to Create PDF/A Compliant Documents

The 3-Heights™ PDF Merge Split API does not have a built-in PDF to PDF/A Converter. This means it can only create PDF/A compliant documents if the input-documents were PDF/A compliant already.

Items that need to be considered when creating PDF/A documents are:

1. Output Intent: Every PDF/A document that uses device specific color spaces (which is the case for most files, even if they only contain text in black color) must have an output intent embedded.
   - The output intent can either be copied from an input file using `CopyOutputIntent` or `CopyAttributes`,
   - or it can be taken from a color profile that resides on the operating system using `OutputIntent`.

   In either case it is required that the output intent is set before any pages are copied from input to output.
2. Metadata: A PDF/A document requires to have XML metadata. There are two ways to set the metadata:
   - Copy the metadata from an existing input document using `CopyMetadata` or `CopyAttributes`.
   - Set them directly using `SetXMPMetadata` or `SetXMPMetadataMem`.
3. Encryption: PDF/A does not allow encryption. In order to prevent encrypting a PDF output document, make sure to:
   - set owner and user password to an empty string, and
   - set the permissions flags to `ePermNoEncryption`

   Code snippet in C#:

```
outdoc.Create("output.pdf", "", "", PDFPermission.ePermNoEncryption);
```

4. Tagging information: PDF/A level A compliance requires the document to be tagged. Make sure to copy tagging information by setting the `ePdfCopyLogicalStructure` flag (see `TPdfCopyOption`) when calling `CopyPages2`
5. Embedded files and associated files: Embedded files can be copied using `CopyEmbeddedFiles`.

For PDF/A-3 compliance, the `ePdfCopyAssociatedFiles` flag must be set (see `TPdfCopyOption`) in order to copy the embedded file's associations in `CopyPages2`.

5.3 How to Efficiently Use the API

5.3.1 How to Copy Pages

The Merge Split API supports copying single pages and page ranges. Prior to copying pages you have to define copy options (`TPdfCopyOption`) that specify what to copy and whether to form fields, links, or other annotations.

Setup copy options:

The following options are recommended for general splitting and merging operations: (See the C include file `pdf-copydecl.h` for exact values of `TPdfCopyOption` as these are not exposed in the COM interface.)

```
Dim options = ePdfCopyAnnotations Or _
  ePdfCopyFormFields Or _
  ePdfCopyLinks Or _
  ePdfCopyLogicalStructure Or _
  ePdfCopyNamedDestinations Or _
  ePdfCopyOutlines Or _
  ePdfCopyAssociatedFiles Or _
  ePdfMergeOCGs Or _
  ePdfSeparateAcroForms
```

Canonical Way for Copying one Document:

```
Dim outdoc As New PDFSPLMRGAPILib.OutDoc
Dim indoc As New PDFSPLMRGAPILib.InDoc
If Not outdoc.Create(...) Then 'do error handling
If Not indoc.Open(...) Then 'do error handling
If Not outdoc.CopyAttributes(indoc) Then 'do error handling
If Not outdoc.CopyPages2(indoc, 1, -1, options) Then 'do error handling
If Not outdoc.CopyEmbeddedFiles(indoc, True) Then 'do error handling
indoc.Close()
outdoc.Close()
```

Efficent use of API:

```
Dim outdoc As New PDFSPLMRGAPILib.OutDoc
Dim indoc As New PDFSPLMRGAPILib.InDoc
outdoc.Create(...)
indoc.Open(...)
outdoc.CopyPages2 indoc, 1, 10, options
indoc.Close()
outdoc.Close()
```

Ineffcent use of API:

```
Dim outdoc As New PDFSPLMRGAPILib.OutDoc
Dim indoc As New PDFSPLMRGAPILib.InDoc
outdoc.Create(...)
indoc.Open(...)
For i = 1 to 10
  outdoc.CopyPages2 indoc, i, i, options
Next i
indoc.Close()
outdoc.Close()
```

The resulting output file is the same. However the first method is much more efficient. Why? Because when copying a page, it's not just the page content that needs to be copied but also data structures in the document's catalogue dictionary. Examples are outlines, form fields, named destinations or the output intent. These data structures have to be processed as a whole for each page range. This happens one time in the first and ten times in the second code snippet.

If performance is crucial and for some reason you have to copy multiple times from the same input to the same output document, you can improve performance by leaving the following flags clear in `TPdfCopyOption` when calling `CopyPages2`:

- `ePdfCopyOutlines`
- `ePdfCopyNamedDestinations`

5.3.2 Operation on Multiple Documents

While working with multiple input and or output documents special care should be taken to keep at any time as few documents open as possible. The Close methods of the input and output documents should be called at the earliest time possible in order to free associated resources.

5.3.3 Features and their Impact on Performance

The performance of the 3-Heights™ PDF Merge Split API is determined by the complexity of the input documents as well as by the options chosen. The following flags of `TPdfCopyOption` have an impact on performance (ordered by effect on performance):

- Tagged PDF: (`ePdfCopyLogicalStructure`)
  Copying and merging of logical structure information is complex and requires both time and memory. You may deactivate this option, if performance is more important to you than preserving tagging information.
- Interactive Form Fields: (`ePdfCopyFormFields`)
  If the input documents contain interactive form fields that need not be editable in the output document, the FlattenFormFields option can be used. This speeds the merge process up significantly while preserving the visual appearance of form fields.
- Optimize Resources: (`ePdfOptimizeResources`)
  Detecting and merging duplicate resources takes both time and memory.
- Outlines: (`ePdfCopyOutlines`)
  Copying and merging of outlines takes some time.
- Named Destinations: (`ePdfCopyNamedDestinations`)
  Copying and merging of named destinations takes both time and memory.

5.4 Error Handling

Most methods of the 3-Heights™ PDF Merge Split API can either succeed or fail depending on user input, state of the PDF Merge Split API, or the state of the underlying system. It is important to detect and handle these errors, to get accurate information about the nature and source of the issue at hand.

Methods communicate their level of success or failure using their return value. Which return values have to be interpreted as failures is documented in the chapter Programmer's Reference. To identify the error on a programmatic level, check the property `ErrorCode`. The property `ErrorMessage` provides a human readable error message, describing the error.

Example:

```
public Boolean Open(string file, string password)
{
  if (!indoc.Open(file, password))
  {
    if (indoc.ErrorCode == PDFErrorCode.PDF_E_PASSWORD)
    {
      password = InputBox.Show("Password incorrect. Enter correct password:");
      return Open(file, password);
    }
    else
    {
      MessageBox.Show(String.Format(
        "Error {0}: {1}", indoc.ErrorCode, indoc.ErrorMessage));
      return false;
    }
  }
  [...]
}
```

6 Programmer's 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 InDoc Interface

6.1.1 Close

Method: `Boolean Close()`

Close an opened input file. If the document is already closed the method does nothing.

Returns:

`True` The file was closed successfully.

`False` Otherwise.

6.1.2 CropBox

Property (get): `Variant CropBox`

This property returns the crop box of the current page defined by the property Page. The crop box rectangle is described by the coordinates left, bottom, right, top. If no crop box is defined, the media box is returned. The values are returned as an array of four single precision real numbers. This property cannot be set.

In order to get the dimensions of the page as it is displayed by a viewer application, the rectangle must be rotated according to the property `Rotate`.

6.1.3 ErrorCode

Property (get): `TPDFErrorCode ErrorCode`

This property can be accessed to receive the latest error code. This value should only be read if a function call on the PDF Merge Split API has returned a value, which signales a failure of the function (see chapter Error Handling) . See also enumeration `TPDFErrorCode`. PDF-Tools error codes are listed in the header file `bseerror.h`. Please note that only few of them are relevant for the 3-Heights™ PDF Merge Split API.

6.1.4 ErrorMessage

Property (get): `String ErrorMessage`

Return the error message text associated with the last error (see property ErrorCode). This message can be used to inform the user about the error that has occurred. For correct usage, see chapter Error Handling.

Note: The property is `Nothing` if no message is available.

6.1.5 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.

6.1.6 GetXMPMetadata

Method: `Boolean GetXMPMetadata(String FileName)`

Write the XMP metadata to the specified file.

6.1.7 GetXMPMetadataMem

Method: `Variant GetXMPMetadataMem()`

Returns XMP metadata as byte-array.

6.1.8 MediaBox

Property (get): `Variant MediaBox`

Use this property to get the PDF "MediaBox" of the current page defined by the property Page. The MediaBox rectangle is described 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`.

6.1.9 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 properties `ErrorCode` and `ErrorMessage` for additional information.

6.1.10 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.

6.1.11 Page

Property (get, set): `Long Page`

This property allows to set and get the currently selected page of an open document given its page number.

The numbers are counted from 1 for the first page to the value of the `PageCount` attribute for the last page. If the document is closed zero is returned.

6.1.12 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`.

6.1.13 Rotate

[Deprecated] Property (get): `Integer Rotate`

Deprecated in Version 4.7 since it has no use anymore.

6.2 OutDoc Interface

6.2.1 AddAssociatedFile

Method: `Boolean AddAssociatedFile(String FileName, String Name, Integer Associate, String AFRelationship, String MimeType, String Description, DATE ModDate)`

Add a file to the document's embedded files. For PDF/A-3, the embedded file is associated with an object of the document, i.e. it is an associated file. This method must be called after `Create` and before `Close`. The file is embedded as-is. Embedding files is not allowed for PDF/A-1 and restricted to PDF/A compliant files for PDF/A-2.

Parameters:

FileName [String] The path (or URL) to the file to be embedded.

Name [String] The name used for the embedded file. This name is presented to the user when viewing the list of embedded files. Default: FileName with the path removed.

Associate [Integer] (Default: -1) The object to associate the embedded file with. -1 for none, 0 for document, number greater than 0 for respective page. If the embedded file is associated with a page, the page must have been copied already.

AFRelationship [String] (Default: "Unspecified") The relationship of the embedded file to the object associate. (Ignored, if Associate is -1.) Allowed values are "Source", "Data", "Alternative", "Supplement", and "Unspecified".

MimeType [String] (Default: "application/octet-stream") Mime-type of the embedded file. Common values other than the default are "application/pdf", "application/xml", or "application/msword".

Description [String] (Default: "") A description of the embedded file. This is presented to the user when viewing the list of embedded files.

ModDate [DATE] The modify date of the file. Default: The modify date of the file on the file system or current time, if not available.

Returns:

True The file was embedded successfully.

False Otherwise.

Example:

```
outdoc.AddAssociatedFile "c:\data\input.doc", "", 0, "Source",
  "application/msword", "", 0
```

6.2.2 AddEmbeddedFile

Method: `Boolean AddEmbeddedFile(String FileName, String Name)`

This is a simplified call that is equal to AddAssociatedFile with default arguments. This is for convenience, for example when embedding files in a PDF/A-2 conforming document.

6.2.3 AddOutlineItem

Method: `AddOutlineItem(String Text, Long PageNo, Single Left, Single Bottom, Single Right, Single Top, Single Zoom, TPDFDestMode Mode)`

This method adds a new outline (bookmark) item including name and its position. To apply multiple outlines, call the function multiple times. The relevant parameters depend on the parameter Mode. The outline item is inserted at the end of the existing outline tree.

Parameters:

`Text` `[String]` The displayed text.

`PageNo` `[Long]` The target page number in this document.

`Left` `[Single]` The left position in points.

`Bottom` `[Single]` The bottom position in points.

`Right` `[Single]` The right position in points.

`Top` `[Single]` The top position in points.

`Zoom` `[Single]` The zoom level; 1 = 100%.

`Mode` `[TPDFDestMode]` The destination type. See `TPDFDestMode`. The parameters `Left`, `Bottom`, `Right`, `Top`, and `Zoom` are only relevant for certain modes.

Example:

```
Dim outdoc As New PDFSPLMRGAPILib.outdoc
Dim indoc As New PDFSPLMRGAPILib.indoc
Dim copyoptions = ePdfCopyAnnotations Or _
    ePdfCopyFormFields Or _
    ePdfCopyLinks Or _
    ePdfCopyLogicalStructure Or _
    ePdfCopyNamedDestinations Or _
    ePdfCopyOutlines Or _
    ePdfCopyAssociatedFiles Or _
    ePdfMergeOCGs

If indoc.Open("in.pdf", "") Then
  outdoc.Create "out.pdf"
  outdoc.CopyOutlines = False
  outdoc.CopyAttributes indoc
  outdoc.CopyPages2 indoc, 1, -1, copyoptions
  outdoc.AddOutlineItem "Page 1, Fit", 1, 0, 0, 0, 0, 0, 1
  outdoc.AddOutlineItem "Page 2, @50,400,300%", 2, 50, 0, 0, 400, 3, 0
  outdoc.AddOutlineItem "Page 3, FitH", 3, 0, 0, 0, 2000, 0, 2
  outdoc.Close
  indoc.Close
End If
Set indoc = Nothing
Set outdoc = Nothing
```

6.2.4 AddOutlineItem2

Method: `AddOutlineItem2(String Text, Long PageNo, Single Left, Single Bottom, Single Right, Single Top, Single Zoom, Integer Mode, Integer Level, Boolean Open)`

This method adds a new outline (bookmark) item including name and its position. To apply multiple outlines, call the function multiple times. The relevant parameters depend on the parameter `Mode`. The outline item is inserted in the end of the existing outline tree at the hierarchy level specified. This method can be used to create an arbitrary outline hierarchy and can be used in combination with the method `CopyOutlineItems2`.

Parameters:

`Text` `[String]` The displayed text.

`PageNo` `[Long]` The target page number in this document.

`Left` `[Single]` The left position in points.

`Bottom` `[Single]` The bottom position in points.

`Right` `[Single]` The right position in points.

`Top` `[Single]` The top position in points.

`Zoom` `[Single]` The zoom level; 1 = 100%.

`Mode` `[Integer]` The destination type. See `TPDFDestMode`. The parameters `Left`, `Bottom`, `Right`, `Top`, and `Zoom` are only relevant for certain modes.

`Level` `[Integer]` The hierarchy level within the existing outline tree at which the outline item is appended. Level `0` for top level.

`Open` `[Boolean]` Whether or not this outline should initially be open (child outline items expanded) or closed (child outline items collapsed).

6.2.5 Author

Property (get, set): `String Author`
Default: `""`

This property sets the Author attribute in the document.

6.2.6 Close

Method: `Boolean Close()`

Close an opened input file. If the document is already closed the method does nothing.

Returns:

`True` The file was closed successfully.

`False` Otherwise.

6.2.7 `CopyAssociatedFiles`

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

Deprecated in Version 4.7. Use `ePdfCopyAssociatedFiles` (see `TPdfCopyOption`) in `CopyPages2`.

6.2.8 `CopyAttributes`

Method: `Boolean CopyAttributes(InDoc InDoc)`

Copy all the document attributes from the given `InDoc`. Calling this method is equivalent with calling the methods `CopyMetadata`, `CopyViewerProperties`, and `CopyOutputIntent`. This method should be called prior to calling `CopyPages2`.

6.2.9 `CopyEmbeddedFiles`

Method: `Boolean CopyEmbeddedFiles(InDoc InDoc, Boolean All)`

This method copies embedded files and associated files from the input document.

Parameters:

`InDoc` `[InDoc]` The input document.

`All` `[Boolean]` If set to `True`: Copy all embedded files. If set to `False`: Copy embedded files associated with document and pages copied with `CopyPages2` only. (PDF/A-3 only, `ePdfCopyAssociatedFiles` flag in `TPdfCopyOption` must be set when calling `CopyPages2`.)

6.2.10 `CopyForms`

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

Deprecated in Version 4.7. Use `ePdfCopyFormFields` and `ePdfFlattenFormFields` (see `TPdfCopyOption`) in `CopyPages2`.

6.2.11 CopyLogicalStructure

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

Deprecated in Version 4.7. Use `ePdfCopyLogicalStructure` (see `TPdfCopyOption`) in `CopyPages2`.

6.2.12 CopyMetadata

Method: `Boolean CopyMetadata(InDoc InDoc)`

Copy info object and XMP metadata from an `InDoc`. This method should only be called once per output document.

Setting the XMP metadata automatically adjusts and thereby overrides the current document info entries. Therefore document info entries must always be applied after setting the XMP metadata to become effective (applies to properties `Author`, `Keywords`, `Subject` and `Title`).

6.2.13 CopyOptionalContent

Method: `Boolean CopyOptionalContent(InDoc InDoc, String Text)`

Copy configuration data for optional content groups (layers). The groups are collected under the name given in the parameter `Text`. If this parameter is `Nothing`, then the document title of `InDoc` is used as `Text`, unless the document title is empty in which case the file name is chosen.

This method is intended to be used prior to calling `CopyPages2` when the `ePdfMergeOCGs` flag is cleared. (See `TPdfCopyOption`.)

6.2.14 CopyOutlines

[Deprecated] Property (get, set): `Boolean CopyOutlines`
Default: `True`

Deprecated in Version 4.7. Use `eCopyOutlines` (see `TPdfCopyOption`) in `CopyPages2` or in `CopyOutlineItems2`.

6.2.15 CopyOutlineItems

[Deprecated] Method: `Boolean CopyOutlineItems(InDoc InDoc, Long FirstPage, Long LastPage, Integer Level)`

Deprecated in Version 4.7. Use `CopyOutlineItems2`.

6.2.16 CopyOutlineItems2

Method: `Boolean CopyOutlineItems2(InDoc InDoc, Long FirstPage, Long LastPage, Integer Level, TPdfCopyOption CopyOptions)`

Copy outline items of the page range. The outline items are inserted in the end of the existing outline tree at the hierarchy level specified. The pages of the page range must be copied using `CopyPages2` before calling the `CopyOutlineItems2` method. The method `CopyOutlineItems2` can be used in combination with the `AddOutlineItem2` method.

Parameters:

`InDoc` [`InDoc`] The input document.

`FirstPage` [`Long`] Specifies the start of the page range in the input document. All outline items belonging to this page range are copied.

`LastPage` [`Long`] Specifies the end of the page range in the input document.

`Level` [`Integer`] Specifies the level hierarchy at which the outline items are inserted. 0 for top level.

`CopyOptions` [`TPdfCopyOption`] Specifies what to copy. (See `TPdfCopyOption`.) For this method, only the following flags are relevant: `ePdfCopyOutlines`, `ePdfCopyLogicalStructure`, `ePdfCopyNamedDestinations`, and `ePdfCopyAssociatedFiles`.

Example:

```
Dim outdoc As New PDFSPLMRGAPILib.outdoc
Dim indoc As New PDFSPLMRGAPILib.indoc
options = ePdfCopyAnnotations Or _
  ePdfCopyFormFields Or _
  ePdfCopyLinks Or _
  ePdfCopyLogicalStructure Or _
  ePdfCopyNamedDestinations Or _
  ePdfCopyAssociatedFiles Or _
  ePdfMergeOCGs
If indoc.Open("in.pdf", "") Then
  outdoc.Create "out.pdf"
  outdoc.CopyPages2 indoc, 1, -1, options
  outdoc.AddOutlineItem2 "in.pdf ", 1, 0, 0, 0, 0, 0, 1, 0, True
  outdoc.CopyOutlineItems2 indoc, 1, -1, options Or ePdfCopyOutlines
  outdoc.Close
  indoc.Close
End If
Set indoc = Nothing
Set outdoc = Nothing
```

6.2.17 `CopyOutputIntent`

Method: `Boolean CopyOutputIntent(InDoc InDoc)`

Copy the PDF/A output intent. This method should only be called once per output document. It should be called prior to copying any pages.

6.2.18 CopyPages

[Deprecated] Method: `Boolean CopyPages(InDoc InDoc, Long FirstPage, Long LastPage)`

Deprecated in Version 4.7. Use `CopyPages2` instead.

6.2.19 CopyPages2

Method: `Boolean CopyPages2(InDoc InDoc, Long FirstPage, Long LastPage, TPDFCopyOptions CopyOptions)`

This method copies a range of pages from the `InDoc`. The method returns `True` if all pages were copied successfully.

Depending on the `CopyOptions` set, outlines, form fields associated with the pages, etc ..., are copied as well. (See `TPdfCopyOption`.)

6.2.20 CopyViewerProperties

Method: `Boolean CopyViewerProperties(InDoc InDoc)`

Copy viewer properties, which include: Page Layout, Page Mode, Open Actions, Piece Info and Collection properties (if `CopyEmbeddedFiles` is used only).

6.2.21 Create

Method: `Boolean Create(String FileName, String UserPassword, String OwnerPassword, Long PermissionFlags)`

Method: `Boolean Create2(String FileName, String UserPassword, String OwnerPassword, Long PermissionFlags, Long KeyLength, String StrF, String StmF)`

Create an output PDF document, apply the security settings and save the content from the input file to the output file.

Note:
- With Version 4.1.13.0 Create2 was added with three new parameters for key length, string filter and stream filter to support AES-V2 and AES-V3 encryption.
- The last three parameters (`KeyLength`, `StrF`, `StmF`) are only relevant in specific cryptographic situations. In all other cases, it is easiest to use the default values `128, "V2", "V2"`.

Parameters:

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

`UserPassword` `[String]` (optional) The user password of the encrypted PDF document.

`OwnerPassword` `[String]` (optional) The owner password of the encrypted PDF document. If the owner password is empty, the user password is used instead.

`PermissionFlags` `[Long]` (optional) Set the permission flags of the PDF document. This option requires an owner password to be set. By default no permissions are granted. The permissions that can be granted are listed in `TPDFPermission`. To not encrypt the output document, set `PermissionFlags` to `-1`, user and `OwnerPassword` to `""`. In order to allow high quality printing, flags `ePermPrint` and `ePermDigitalPrint` need to be set.

`KeyLength` `[Long]` (Default: `128`) The key length is a determining factor of the strength of the encrypting algorithm and the amount of time to break the cryptographic system. For RC4 the key length can be any value from 40 to 128 that is a multiple of 8. For AESV2 the key length is automatically set to 128, for AESV3 to 256.

> **Note:**
> - Certain PDF viewers only support 40 and 128 bit encryption. Other tools, such as the 3-Heights™ tools also support other encryption key lengths.
> - 256 bit encryption requires Acrobat 9 or later.

`StrF` `[String]` (Default: `"V2"`) Set the string crypt filter. Setting this value to an empty string or `Nothing` means the default filter is used. Supported crypt filters:

`"None"` The application does not decrypt data.

`"V2"` (PDF 1.2) The application asks the security handler for the encryption key and implicitly decrypts data using the RC4 algorithm.

`"RC4"` Same as `"V2"`

`"AESV2"` (PDF 1.6) The application asks the security handler for the encryption key and implicitly decrypts data using the AES-V2 128 bit algorithm.

`"AESV3"` (PDF 1.7) The application asks the security handler for the encryption key and implicitly decrypts data using the AES-V3 256 bit algorithm.

`StmF` `[String]` (Default: `"V2"`) Set the stream crypt filter. Supported values are `"None"`, `"V2"`, `"RC4"`, `"AESV2"` and `"AESV3"`. Note that certain viewers require the stream crypt filter to be equal to the string crypt filter, e.g. both must be RC4 or AES. Other tools, such as the 3-Heights™ PDF tools do not have this limitation. Setting this value to an empty string or `Nothing`, means the default filter is used.

Returns:

`True` The file was created successfully.

`False` The file was not created. This can be due to permissions or a locked file, or another reason. See also `ErrorCode` and `ErrorMessage`.

6.2.22 CreateInMemory

Method: `Boolean CreateInMemory()`

Method: `Boolean CreateInMemory2(String UserPassword, String OwnerPassword, Long PermissionFlags, Long KeyLength, String StrF, String StmF)`

This method saves the output PDF in memory as a byte array. (See also method `GetPdf`.) For a description of the parameters of `CreateInMemory2`, see method `Create`.

Returns:

`True` The PDF document was created successfully.

`False` The PDF document was not created successfully. See also `ErrorCode` and `ErrorMessage`.

6.2.23 ErrorCode

Property (get): `TPDFErrorCode ErrorCode`

This property can be accessed to receive the latest error code. This value should only be read if a function call on the PDF Merge Split API has returned a value, which signales a failure of the function (see chapter Error Handling) . See also enumeration `TPDFErrorCode`. PDF-Tools error codes are listed in the header file `bseerror.h`. Please note that only few of them are relevant for the 3-Heights™ PDF Merge Split API.

6.2.24 ErrorMessage

Property (get): `Boolean ErrorMessage`
Default: `False`

Return the error message text associated with the last error (see property `ErrorCode`). Note that the property is `Nothing` if no message is available.

6.2.25 FlattenAnnotations

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

Deprecated in Version 4.7. Use `ePdfFlattenAnnotations` (see `TPdfCopyOption`) in `CopyPages2`.

6.2.26 FlattenFormFields

[Deprecated] **Property (get, set):** `Boolean FlattenFormFields`
Default: `False`

Deprecated in Version 4.7. Use `ePdfFlattenFormFields` (see `TPdfCopyOption`) in `CopyPages2`.

6.2.27 FlattenSigAppearance

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

Deprecated in Version 4.7. Use `ePdfFlattenAnnotations` (see `TPdfCopyOption`) in `CopyPages2`.

6.2.28 GetPdf

Method: `Variant GetPdf()`

Get the output file from memory. See also method `CreateInMemory`.

Returns:

A byte array containing the output PDF. In certain programming languages, such as Visual Basic 6, the type of the byte array must explicitly be Variant.

6.2.29 InfoEntry

Method: `String InfoEntry(String Key)`

Retrieve or add a key-value pair to the document info dictionary. Values of predefined keys are also stored in the XMP metadata package.

Popular entries specified in the PDF Reference 1.7 and accepted by most PDF viewers are `"Title"`, `"Author"`, `"Subject"`, `"Creator"` (sometimes referred to as Application) and `"Producer"` (sometimes referred to as PDF Creator).

Parameter:

`Key`	`[String]`	A key as string.

Returns:

The value as string.

Examples in Visual Basic 6:

Get the document title.

```
t = doc.InfoEntry("Title")
```

Set the document title.

```
doc.InfoEntry("Title") = "My Title"
```

Set the creation date to 13:55:33, April 5, 2010, UTC+2.

```
doc.InfoEntry("CreationDate") = "D:20100405135533 + 02'00'"
```

6.2.30 Keywords

Property (get, set): `String Keywords`
Default: `""`

Add keywords to the document or retrieve keywords of the document.

6.2.31 LicenseIsValid

Property (get): `Boolean LicenseIsValid`

Check if the license is valid.

6.2.32 Linearize

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

Get or set whether to linearize the PDF output file, i.e. optimize file for fast web access.

A linearized document has a slightly larger file size than a non-linearized file and provides the following main features:

- When a document is opened in a PDF viewer of a web browser, the first page can be viewed without downloading the entire PDF file. In contrast, a non-linearized PDF file must be downloaded completely before the first page can be displayed.
- When another page is requested by the user, that page is displayed as quickly as possible and incrementally as data arrives, without downloading the entire PDF file.

The above applies only if the PDF viewer supports fast viewing of linearized PDFs.

When enabling this option, then no PDF objects will be stored in object streams in the output PDF. For certain input documents this can lead to a significant increase of file size.

6.2.33 MergeOptionalContent

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

Deprecated in Version 4.7. Use `ePdfMergeOCGs` (see `TPdfCopyOption`) in `CopyPages2`.

6.2.34 OptimizeResources

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

Deprecated in Version 4.7. Use `ePdfOptimizeResources` (see `TPdfCopyOption`) in `CopyPages2`.

6.2.35 OutputIntent

Property (set): `String OutputIntent`
Default: `""`

Load the PDF/A output intent's color profile from specified file.

6.2.36 PageLayout

Property (set): `TPDFPageLayout PageLayout`

Set the page layout that shall be used when the document is opened. Alternatively the page layout can be copied from an input document using the method `CopyViewerProperties`. See `TPDFPageLayout` for an explanation of the different page layouts.

6.2.37 PageMode

Property (set): `TPDFPageMode PageMode`

Set the page mode that specifies how the document shall be displayed when opened. Alternatively the page mode can be copied from an input document using the method `CopyViewerProperties`. See `TPDFPageMode` for an explanation of the different page modes.

6.2.38 ProductVersion

Property (get): `String ProductVersion`

Get the version of the 3-Heights™ PDF Merge Split API in the format "A.C.D.E".

6.2.39 RemoveNamedDests

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

Deprecated in Version 4.7. Use `ePdfCopyNamedDestinations` (see `TPdfCopyOption`) in `CopyPages2`. If this property is set, all named destinations of the input document are removed and all internal named destinations converted to regular destinations.

6.2.40 Rotate

Property (get, set): `Integer Rotate`
Default: `0`

The number of degrees by which the page should be rotated additionally when the document is viewed (or printed). This value must be set before copying pages to this output document with `CopyPages2`. A positive value is a clockwise rotation. The value must be a multiple of `90`. I.e. valid values are `-270`, `-180`, `-90`, `0`, `90`, `180`, `270`. The default is `0`.

6.2.41 SetLicenseKey

Method: `Boolean SetLicenseKey(String LicenseKey)`

Set the license key.

6.2.42 SetOpenAction

Method: `Boolean SetOpenAction(Long PageNo, Single Left, Single Bottom, Single Right, Single Top, Single Zoom, TPDFDestMode Mode)`

The open action defines which page shall be presented to the user initially upon opening the document.

Parameters:

PageNo `[Long]` The target page number.

Left `[Single]` The left position in points.

Bottom `[Single]` The bottom position in points.

Right `[Single]` The right position in points.

Top `[Single]` The top position in points.

Zoom `[Single]` The zoom level; 1 = 100%.

Mode `[TPDFDestMode]` The destination type. See `TPDFDestMode`.

6.2.43 SetViewerPreference

Method: `Boolean SetViewerPreference(String Key, String Value)`

Add an entry to the viewer preferences dictionary. All properties defined in the PDF Reference 1.7 and earlier are supported.

Parameters:

Key [String] The name of the entry.

Value [String] A string representation of the value. Names: value string, Booleans: "true" or "false", Integers: decimal numbers like "22", Arrays: comma-separated list of items like "1, 3, 55"

6.2.44 SetXMPMetadata

Method: `Boolean SetXMPMetadata(String FileName)`

Load XMP metadata from specified file. Setting the XMP metadata automatically adjusts and thereby overrides the current document info entries. Therefore document info entries must always be applied after setting the XMP metadata to become effective (applies to properties "Author", "Keywords", "Subject" and "Title").

6.2.45 SetXMPMetadataMem

Method: `Boolean SetXMPMetadataMem(Variant Mem)`

Load XMP metadata from a byte array. See also SetXMPMetadata.

6.2.46 Subject

Property (get, set): `String Subject`
Default: ""

This property sets the Subject attribute of the document.

6.2.47 Title

Property (get, set): `String Title`
Default: ""

This property sets the Title attribute of the document.

6.3 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).

6.3.1 TPdfCopyOption

The recommended default value for CopyOptions is:

```
options = ePdfCopyAnnotations Or _
          ePdfCopyFormFields Or _
          ePdfCopyLinks Or _
          ePdfCopyLogicalStructure Or _
          ePdfCopyOutlines Or _
          ePdfCopyAssociatedFiles Or _
          ePdfMergeOCGs
```

When creating PDF/A level A documents, e.g. PDF/A-2a, or if universal accessbility is important, also add ePdf-CopyLogicalStructure.

TPDFCopyOption

TPDFCopyOption	Description
ePdfCopyAnnotations	Copy interactive annotations such as sticky notes or highlight annotations.
ePdfCopyFormFields	Copy interactive form fields. Note that when merging multiple documents with form fields, it is important that no two different form fields have the same name. Otherwise one of the fields might have to be renamed (see ePdfSeparateAcroForms). Consider using ePdfFlattenFormFields when merging multiple forms.
ePdfCopyLinks	Copy links (document internal and external links).
ePdfCopyLogicalStructure	Copy logical structure information. Logical structure information in a PDF defines the structure of content, such as titles, paragraphs, figures, reading order, tables or articles. Logical structure elements can be “tagged” with descriptions or alternative text. E.g. “tagging” allows the contents of an image to be described to the visually impaired. It is recommended to use this option, if all input documents are “tagged”. Otherwise this could be deactivated in order to create smaller output files and get a much better performance. This option is required for PDF/A level A compliance (e.g. PDF/A-1a, PDF/A-2a, PDF/A-3a).

TPDFCopyOption

`ePdfCopyNamedDestinations`	Copy named destinations. A document may contain a mapping of names to destinations within the document. These names can then be used in link annotations or outlines in order to refer to destinations within the document. Links within the document will work regardless of the state of this flag. If `ePdfCopyNamedDestinations` is not used, all named destinations of the input document are removed and all internal named destinations converted to regular destinations. This is much faster than copying named destinations. If a document is split into multiple documents with the intention of merging the pieces back together at a later time, this flag should be used. If the document uses named destinations, links between the pieces will work after merging if `ePdfCopyNamedDestinations` is used.
`ePdfCopyOutlines`	Copy all outline items (bookmarks) that point to the copied pages. The structure of the outline tree in the output document will be the same as in the input document, regardless of the order in which pages are copied.
`ePdfFlattenAnnotations`	Flatten annotations preserves the visual appearance of annotations, but discards all interactive elements. When using this option, it is recommended to leave the `ePdfCopyAnnotations` flag cleared. Note that this option does not flatten form fields, signature appearances and links, even though technically these are annotations as well.
`ePdfFlattenFormFields`	Flatten form fields preserves the visual appearance of form fields, but discards all interactive elements. When using this option, it is recommended to leave the `ePdfCopyFormFields` flag cleared. Often, form fields have no associated visual appearance stored in the document. For such fields an appearance must be generated when flattening. The 3-Heights™ PDF Merge Split API currently cannot generate an appearance for all types of form fields. If an appearance generation failed then an `ErrorCode` is set. See also `TPDFErrorCode`.

TPDFCopyOption

`ePdfFlattenSignatureApprearances`	Flatten the visual appearance of signed signature fields. A digital signature consists of two parts: First, a cryptographic part that includes a hash value based on the content of the document that is being signed. If the document is modified at a later time, the computed hash value is no longer correct and the signature becomes invalid, i.e. the validation will fail and will report that the document has been modified since the signature has been applied. Second, an optional visual appearance on a page of the PDF document. The signature appearance can be useful to indicate the presence of a digital signature by a particular signer. Processing the PDF with 3-Heights™ PDF Merge Split API breaks the signature, and therefore the cryptographic part needs to be removed. In general, the visual appearance is regarded as worthless without the cryptographic part, it is removed by default. The visual appearance can be preserved by setting the flag `ePdfFlattenSignatureAppearances`.
`ePdfOptimizeResources`	Find and merge redundant resources from different input files. Equal fonts, images and color spaces are detected. By activating this feature, much smaller output files are created, if similar files are merged. However, the merging process uses more time and memory resources.
`ePdfCopyAssociatedFiles`	Copy associated files. For PDF/A-3 compliance, this option must be set and the method `CopyEmbeddedFiles` must be called.
`ePdfMergeOCGs`	Merge compatible optional content groups (layers). If this option is set the configuration of optional content is compared with the input file. If it is found to be the same then the optional content groups are assumed to be the same in the input and the output document and merging takes place. If they are different then optional content groups are assumed to be distinct and they are simply added. If this option is not set then no configuration of optional content groups is copied. In this case one can use the method `CopyOptionalContent` to copy such configuration information.

TPDFCopyOption

ePdfSeparateAcroForms	Keep AcroForm fields from different files separate even if they are identical. This option has only an effect, if ePdfCopyFormFields is used. AcroForm fields are key-value pairs. The key (name) is important if for example the form's content is submitted to a web server or modified using JavaScript actions embedded within the document. For most forms it is therefore crucial that the name of form fields is preserved. This option controls the behavior of the 3-Heights™ PDF Merge Split API when merging files that contain form fields with the same name. If this option is set, fields are renamed if the output document already contains a field of the same name. If this option is not set, fields are merged into one field if they are of the same type and have the same value. Otherwise fields are renamed. Note that if two fields are merged, multiple widget annotations (i.e. the appearance of the field on pages) will be associated with the same form field and therefore will always show the same value.

6.3.2 TPDFDestMode

A PDF destination defines a particular view of the document including the page, the location on the page and the zoom factor. There are eight different modes (enumerated by TPDFDestMode) to specify the location on the page and the zoom factor. Depending on the mode, between 0 and 4 parameters are required to define the destination.

TPDFDestMode Table

Value	Parameters	Description
eDestModeXYZ	left top zoom	The upper left corner of the view is positioned at the coordinate (left, top) with the given zoom factor.
eDestModeFit		The view is such that the whole page is visible.
eDestModeFitH	top	The view is top-aligned with top and shows the whole page width.
eDestModeFitV	left	The view is left-aligned with left and shows the whole page height.
eDestModeFitR	left bottom right top	The view contains the rectangle specified the two coordinates (left, bottom) and (right, bottom).
eDestModeFitB		The view is such that the pages bounding box is visible.

TPDFDestMode Table

eDestModeFitBH top	The view is top-aligned with top and shows the whole width of the page's bounding box.
eDestModeFitBV left	The view is left-aligned with left and shows the whole height of the page's bounding box.

For more information about PDF destinations please see Chapter 8.2.1 in the PDF Reference 1.7.

6.3.3 TPDFErrorCode

All TPDFErrorCode enumerations start with a prefix, such as PDF_, followed by a single letter which is one of S, E, W or I, an underscore and a descriptive text.

The single letter gives an indication of the severity of the error. These are: Success, Error, Warning and Information. In general, an error is returned if an operation could not be completed, e.g. no valid output file was created. 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.PDFError-Code. Note that only a few are relevant for the 3-Heights™ PDF Merge Split API, most of which are listed here:

TPDFErrorCode Table

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

TPDFErrorCode Table

PDF_SPLMRG_W_RMXFA	XFA stream was not copied.
PDF_SPLMRG_W_RMSUBMIT	SubmitForm action was not copied.
PDF_SPLMRG_W_PARTSUBMIT	Partial SubmitForm action altered to submit all fields.
PDF_SPLMRG_W_RMSIGANNOT	Signature annotation was not copied.
PDF_SPLMRG_W_RMVALUE	Value or default value of a field was discarded due to field name collision.
PDF_SPLMRG_W_MVFIELD	Renamed a form field due to field name collision.
PDF_SPLMRG_E_ANNOTAPPEAR	Failed to generate an appearance for a form field.

6.3.4 TPDFPermission

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

TPDFPermission Table

TPDFPermissionFlag	Description
ePermNoEncryption	Do not apply encryption. This enumeration shall not be combined with another enumeration. When using this enumeration set both passwords to an empty string or Nothing.
ePermNone	Grant no permissions
ePermPrint	Low resolution printing
ePermModify	Changing the document
ePermCopy	Content copying or extraction
ePermAnnotate	Annotations
ePermFillForms	Filling of form fields
ePermSupportDisabilities	Support for disabilities
ePermAssemble	Document assembly
ePermDigitalPrint	High resolution printing
ePermAll	Grant all permissions

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

Allow Printing

```
Permission = Permission Or ePermPrint
```

Prohibit Printing

```
Permission = Permission And Not ePermPrint
```

6.3.5 TPDFPageLayout

The page layout defines how the document's pages are displayed when it is opened. There are six different layouts (enumerated by `TPDFPageLayout`) to specify which content of the document is visible when it is opened.

TPDFPageLayout Table

PageLayout	Description
`ePageLayoutSinglePage`	One page is displayed at a time.
`ePageLayoutOneColumn`	Pages are displayed in one column.
`ePageLayoutTwoColumnLeft`	Pages are displayed in two columns, with oddnumbered pages on the left.
`ePageLayoutTwoColumnRight`	Pages are displayed in two columns, with oddnumbered pages on the right.
`ePageLayoutTwoPageLeft`	Two pages are displayed at a time, with oddnumbered pages on the left.
`ePageLayoutTwoPageRight`	Two pages are displayed at a time, with oddnumbered pages on the right.

For more information please see Chapter 3.6.1 in the PDF Reference 1.7.

6.3.6 TPDFPageMode

The page mode defines how the document is displayed when it is opened. There are six different modes (enumerated by `TPDFPageMode`) to specify which content of the document is visible when it is opened.

TPDFPageMode Table

PageMode	Description
`ePageModeUseNone`	Neither document outline nor thumbnail images are displayed.
`ePageModeUseOutlines`	The document outline is visible.

TPDFPageMode Table

`ePageModeUseThumbs`	Thumbnail images are visible.
`ePageModeFullScreen`	The document is displayed in full-screen mode. The menu bar, window controls, or any other windows are not visible.
`ePageModeUseOC`	The optional content group panel is visible.
`ePageModeUseAttachments`	The attachment panel appears.

For more information please see Chapter 3.6.1 in the PDF Reference 1.7.

7 Examples

Examples in various programming languages are included in the ZIP files of the release and evaluation versions.

8 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.

8.1 Changes in Version 4.10

- **Changed** the behavior when copying outlines. The outline structure in the output file now always matches the outline structure in the input file, regardless of the order in which pages are copied.
- Improved reparation of corrupt form fields.
- Writing PDF objects into object streams is now supported. Most objects that are contained in object streams in the input document are now also stored in object streams in the output document. When enabling linearization, however, no objects are stored in object streams.
- Increased robustness against corrupt input PDF documents.
- [C] **Clarified** Error handling of `TPdfStreamDescriptor` functions.

8.2 Changes in Version 4.9

- Improved support for and robustness against corrupt input PDF documents.
- Improved repair of embedded font programs that are corrupt.
- Improved metadata generation for standard PDF properties.
- [C] **Changed** return value `pfGetLength` of `TPDFStreamDescriptor` to `pos_t`[5].
- [PHP] **New** Interface for Windows and Linux. Supported versions are PHP 5.6 & 7.0 (Non Thread Safe). The `PdfSplMrgAPI` PHP Interface is contained in the 3-Heights™ PDF Tools PHP5.6 Extension and the 3-Heights™ PDF Tools PHP7.0 Extension.

8.3 Changes in Version 4.8

- **New** warning issued if input page range is outside of the input document's pages.
- The creation of annotation appearances has been optimized to use less memory and processing time.
- Added repair functionality for TrueType font programs whose glyphs are not ordered correctly.

Interface `OutDoc`

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

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

9 Licensing, Copyright, and Contact

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

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

Licensing and Copyright

Contact

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