User Manual

# 3-Heights®
# PDF Merge Split API

PDF/A conform

**Version 6.27.2**

pdftools

# Contents

# 1 Introduction

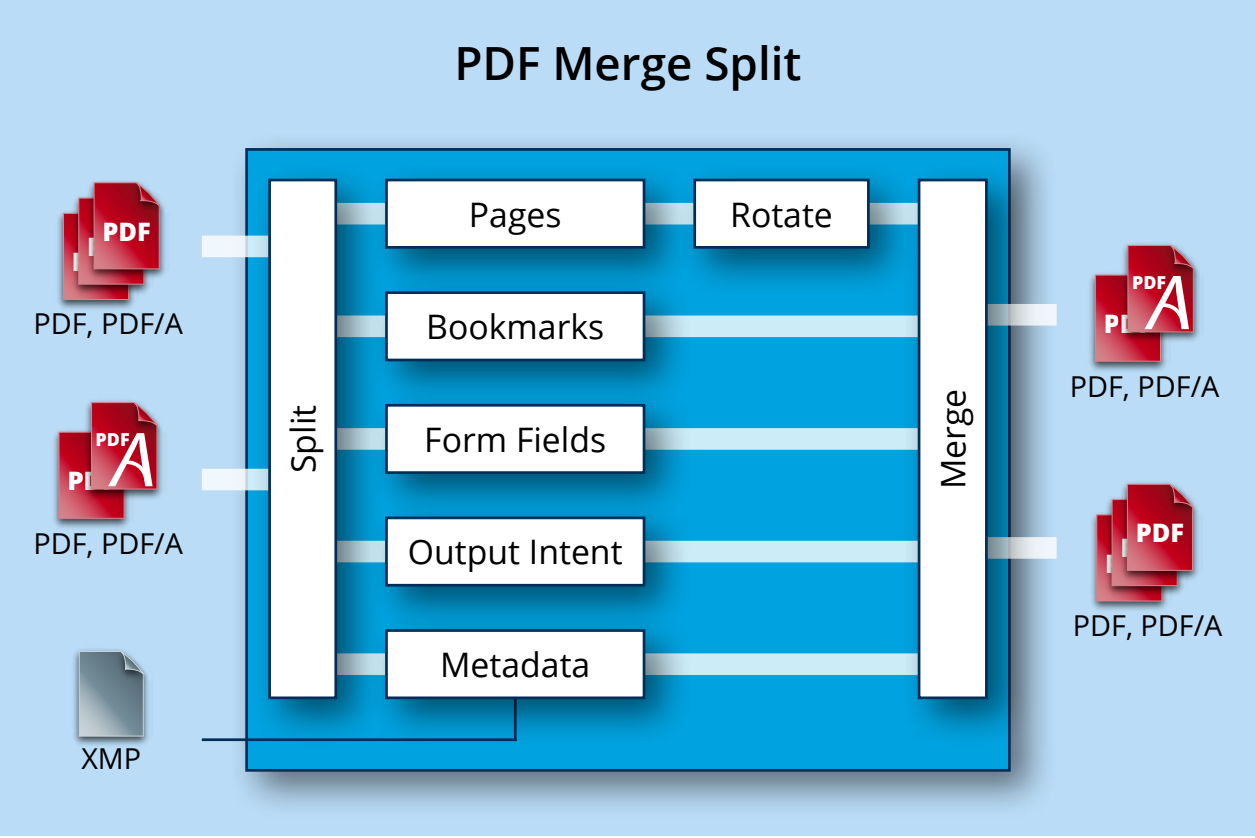
The 3-Heights® PDF Merge Split API is a component for splitting and merging the pages of PDF documents with useful additional functions.

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 flatten form fields.

A special feature is the component's ability to process and create PDF/A conforming files.

## 1.1 Functions

The 3-Heights® PDF Merge Split API can operate on multiple input and output documents in one processing step.




### 1.1.1 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 with automatically chosen version and conformance level (down-grade).
- Automatic PDF version upgrade when merging documents with differing PDF version. Merging PDF 1.x and PDF 2.0 is currently not supported.
- 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) (not PDF 2.0)
- Set the page mode, initial page layout, and open action
- Split vertical or horizontal double pages into single pages

### Input formats

- PDF 1.x (PDF 1.0, …, PDF 1.7)
- PDF 2.0
- PDF/A-1, PDF/A-2, PDF/A-3

### Output formats

- PDF 1.x (PDF 1.0, …, PDF 1.7)
- PDF 2.0
- PDF/A-1, PDF/A-2, PDF/A-3

### Conformance

Standards:

- ISO 32000-1 (PDF 1.7)
- ISO 32000-2 (PDF 2.0)
- ISO 19005-1 (PDF/A-1)
- ISO 19005-2 (PDF/A-2)
- ISO 19005-3 (PDF/A-3)

## 1.2 Interfaces

The following interfaces are available

- C
- .NET
- Java
- COM

## 1.3 Operating systems

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

- Windows Client 7+ | x86 and x64
- Windows Server 2008, 2008 R2, 2012, 2012 R2, 2016, 2019, 2022 | x86 and x64
- Linux:
  - Red Hat, CentOS, Oracle Linux 7+ | x64
  - Fedora 29+ | x64
  - Debian 8+ | x64
  - Other: Linux kernel 2.6+, GCC toolset 4.8+ | x64
- macOS 10.10+ | x64

'+' indicates the minimum supported version.

## 1.4 How to best read this manual

If you are reading this manual for the first time and would like to evaluate the software, the following steps are suggested:

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

# 2 Installation and deployment

## 2.1 Windows

The 3-Heights® PDF Merge Split API comes as a ZIP archive or as a NuGet package.

To install the software, proceed as follows:

1. You need administrator rights to install this software.
2. Log in to your download account at https://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 can find different versions of the product available. Download the version that is selected by default. You can select a different version.
   The product comes as a ZIP archive containing all files, or as a NuGet package containing all files for development in .NET.
   There is a 32 and a 64-bit version of the product available. While the 32-bit version runs on both 32 and 64-bit platforms, the 64-bit version runs on 64-bit platforms only. The ZIP archive as well as the NuGet package contain both the 32-bit and the 64-bit version of the product.
3. If you are using the ZIP archive, uU nzip the archive to a local folder, e.g. `C:\Program Files\PDF Tools AG\`.
   This creates the following subdirectories (see also ZIP archive):

| Subdirectory | Description |
|---|---|
| `bin` | Runtime executable binaries |
| `doc` | Documentation |
| `include` | Header files to include in your C/C++ project |
| `jar` | Java archive files for Java components |
| `lib` | Object file library to include in your C/C++ project |
| `samples` | Sample programs in various programming languages |

4. The usage of the NuGet package is described in section NuGet package.
5. (Optional) Register your license key using the License management.
6. Identify the interface you are using. Perform the specific installation steps for that interface described in Interface-specific installation steps.

## 2.2 Linux and macOS

This section describes installation steps required on Linux or macOS.

The Linux and macOS version of the 3-Heights® PDF Merge Split API provides two interfaces:

- Java interface
- Native C interface

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

File description

| Name | Description |
|---|---|
| bin/x64/libPdfSplMrgAPI.so | Shared library that contains the main functionality. The file's extension differs on macOS, (.dylib instead of .so). |
| doc/*.* | Documentation |
| include/*.h | Header files to include in your C/C++ project |
| jar/MSPA.jar | Java API archive |
| samples | Example code |

### 2.2.1 Linux

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

```
ldd libPdfSplMrgAPI.so
```

   If the previous step reports any missing libraries, you have two options:
   a. Download an archive that is linked to a different version of the GNU shared libraries and verify whether they are available on your system. Use any version whose requirements are met. Note that this option is not available for all platforms.
   b. Use your system's package manager to install the missing libraries. It usually suffices to install the package libstdc++6.
3. Create a link to the shared library from one of the standard library directories, e.g.

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

4. Optionally, register your license key using the license manager.
5. Identify the interface you are using. Perform the specific installation steps for that interface described in Interface-specific installation steps.

### 2.2.2 macOS

The shared library must have the extension .jnilib for use with Java. Create a file link for this purpose by using the following command:

```
ln libPdfSplMrgAPI.dylib libPdfSplMrgAPI.jnilib
```

## 2.3 ZIP archive

The 3-Heights® PDF Merge Split API provides four different interfaces. The installation and deployment of the software depend on the interface you are using. The table below shows the supported interfaces and some of the programming languages that can be used.

| Interface | Programming languages |
|---|---|
| .NET | The MS software platform .NET can be used with any .NET capable programming language such as:<br>■ C#<br>■ VB .NET<br>■ J#<br>■ others<br>For a convenient way to use this interface, see NuGet package. |
| 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:<br>■ MS Visual Basic<br>■ MS Office Products such as Access or Excel (VBA)<br>■ C++<br>■ VBScript<br>■ others<br>This interface is available in the Windows version only. |
| C | The native C interface is for use with C and C++. This interface is available on all platforms. |

### 2.3.1 Development

The software development kit (SDK) contains all files that are used for developing the software. The role of each file in each of the four different interfaces is shown in table Files for development. The files are split in four categories:

**Req.** The file is required for this interface.

**Opt.** The file is optional. See also the File description table to identify the files are required for your application.

**Doc.** The file is for documentation only.

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

**Files for development**

| Name | .NET | Java | COM | C |
|---|---|---|---|---|
| bin\‹platform›\PdfSplMrgAPI.dll | Req. | Req. | Req. | Req. |
| bin*NET.dll | Req. | | | |
| bin*NET.xml | Doc. | | | |
| doc*.pdf | Doc. | Doc. | Doc. | Doc. |
| doc\PdfSplMrgAPI.idl | | | Doc. | |
| doc\javadoc*.* | | Doc. | | |

**Files for development**

| Name | .NET | Java | COM | C |
|---|---|---|---|---|
| include\pdfsplmrgapi_c.h | | | | Req. |
| include*.* | | | | Opt. |
| jar\MSPA.jar | | Req. | | |
| lib\‹platform›\PdfSplMrgAPI.lib | | | | Req.[1] |
| samples*.* | Doc. | Doc. | Doc. | Doc. |

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

**File description**

| Name | Description |
|---|---|
| bin\‹platform›\PdfSplMrgAPI.dll | DLL that contains the main functionality (required), where ‹platform› is either Win32 or x64 for the 23-bit or the 64-bit library, respectively. |
| bin*NET.dll | .NET assemblies are required when using the .NET interface. The files bin*NET.xml contain the corresponding XML documentation for MS Visual Studio. |
| doc*.* | Documentation |
| include*.* | Files to include in your C / C++ project |
| lib\‹platform›\PdfSplMrgAPI.lib | On Windows operating systems, the object file library needs to be linked to the C/C++ project. |
| jar\MSPA.jar | Java API archive |
| samples*.* | Sample programs in different programming languages |

### 2.3.2 Deployment

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

**Files for deployment**

| Name | .NET | Java | COM | C |
|---|---|---|---|---|
| bin\‹platform›\PdfSplMrgAPI.dll | Req. | Req. | Req. | Req. |

[1] Not required for Linux or macOS.
[2] These files must reside in the same directory as PdfSplMrgAPI.dll.

**Files for deployment**

| | |
|---|---|
| `bin\*NET.dll` | Req. |
| `jar\MSPA.jar` | Req. |

The deployment of an application works as described below:

1. Identify the required files from your developed application (this may also include color profiles).
2. Identify all files that are required by your developed application.
3. Include all these files in an installation routine such as an MSI file or a 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 PowerUser privileges and is added to the batch script.

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

## 2.4 NuGet package

NuGet is a package manager that lets you integrate libraries for software development in .NET. The NuGet package for the 3-Heights® PDF Merge Split API contains all the libraries needed, both managed and native.

### Installation

The package `PdfTools.PdfSplMrg 6.27.2` is available on nuget.org. Right-click on your .NET project in Visual Studio and select “Manage NuGet Packages...”. Finally, select the package source “nuget.org” and navigate to the package `PdfTools.PdfSplMrg 6.27.2`.

### Development

The package `PdfTools.PdfSplMrg 6.27.2` contains .NET libraries with versions .NET Standard 1.1, .NET Standard 2.0, and .NET Framework 2.0, and native libraries for Windows, macOS, and Linux.

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

To use the software, you must first install a license key for the 3-Heights® PDF Merge Split API. To do this, you have to download the product kit and use the license manager in it. See also License management.

**Note:** This NuGet package is only supported on a subset of the operating systems supported by .NET Core. See also Operating systems.

## 2.5 Interface-specific installation steps

### 2.5.1 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 how to register the PdfSplMrgAPI.dll. 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 `%SystemRoot%\System32`.[3]

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`.

### 2.5.2 Java interface

The 3-Heights® PDF Merge Split API requires Java version 7 or higher.

#### For compilation and execution

When using the Java interface, the Java wrapper `jar\MSPA.jar` needs to be on the CLASSPATH. You can do this 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" ^
 sampleApplication.java
```

#### For execution

Additionally, the library `PdfSplMrgAPI.dll` needs be in one of the system's library directories[4] or added to the Java system property `java.library.path`. You can add the library by either adding it dynamically at program startup before using the API, or by specifying it using the switch `-Djava.library.path` when starting the Java VM. Choose the correct subdirectory (`x64` or `Win32` on Windows) depending on the platform of the Java VM[5].

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

[4] On Windows defined by the environment variable PATH, and on Linux defined by `LD_LIBRARY_PATH`.

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

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

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

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

### 2.5.3 .NET interface

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

It is recommended that you use the NuGet package. This ensures the correct handling of both the .NET assemblies and the native library.

Alternatively, the files in the ZIP archive can be used directly in a Visual Studio project targeting .NET Framework 2.0 or later. To achieve this, proceed as follows:

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

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

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

In Visual Studio, when using the platforms "x86" or "x64", you can do this by adding the 32-bit or 64-bit `PdfSplMrgAPI.dll`, respectively, as an "existing item" to the project, and setting its property "Copy to output directory" to true. When using the "AnyCPU" platform, make sure, by some other means, that both the 32-bit and the 64-bit `PdfSplMrgAPI.dll` are copied to subdirectories `win-x86` and `win-x46` of the output directory, respectively.

### 2.5.4 C interface

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

## 2.6 Uninstall, Install a new version

If you have used the ZIP file for the installation, undo all the steps done during installation, e.g. de-register using `regsvr32.exe /u`, delete all files, etc.

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

## 2.7 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

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

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

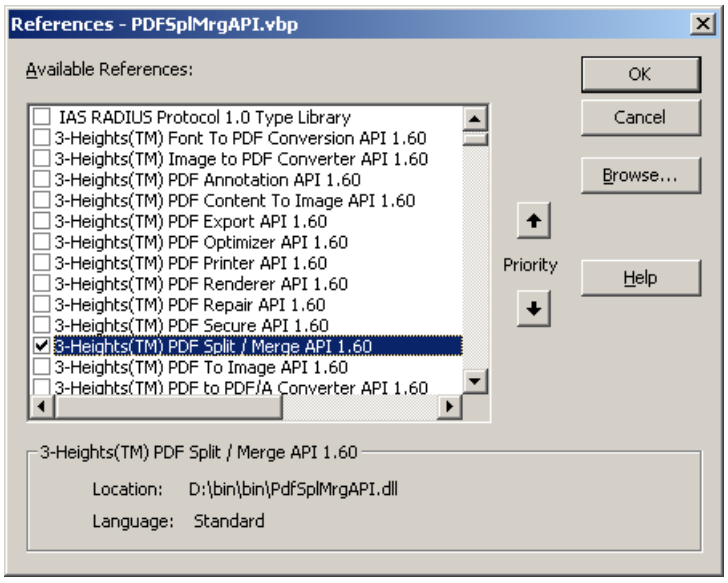
# 4 Programming interfaces

## 4.1 Visual Basic 6

After installing the 3-Heights® PDF 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, perform these steps:

1. Create a new Standard-Exe Visual Basic 6 project. Then include the 3-Heights® PDF Merge Split API component to your project.




2. Draw a new Command Button and optionally, rename it as appropriate.
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 to quickly start is to refer to this sample.

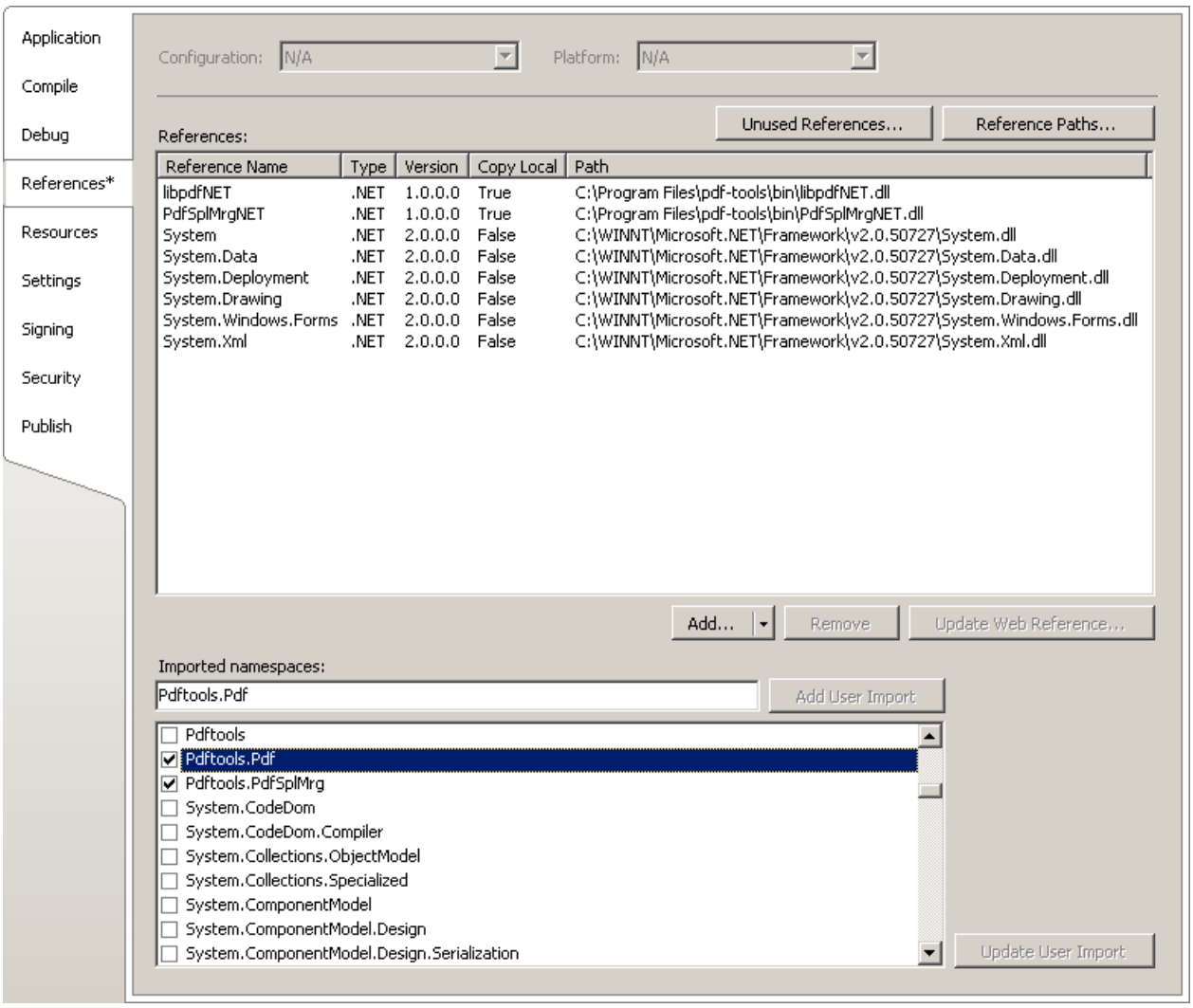
To create a new project from scratch, perform the following steps:

1. Start Visual Studio and create a new C# or VB project.
2. Add references to the NuGet package `PdfTools.PdfSplMrg 6.27.2`, as described in NuGet package.
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:




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. See also .NET interface.
2. For deployment, all items in the project's output directory (e.g. `bin\Release`) must be copied to the target computer. This includes the 3-Heights® PDF Merge Split API's .NET assemblies (`*NET.dll`), as well as the native library (`PdfSplMrgAPI.dll`) in its 32 bit or 64 bit version or both. The native library can alternatively be copied to a directory listed in the PATH environment variable, e.g. `%SystemRoot%\System32`.
3. It is crucial that the native library `PdfSplMrgAPI.dll` is found at execution time, and that the native library's format (32 bit versus 64 bit) matches the operating system.
4. The output directory may contain multiple versions of the native library, e.g. for Windows 32 bit, Windows 64 bit, MacOS 64 bit, and Linux 64 bit. Only the versions that match the target computer's operating system need be deployed.
5. If required by the application, optional DLLs must be copied to the same folder. See Deployment for a list and description of optional DLLs.

### 4.2.4 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, which you distinguish 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 these issues.

#### Troubleshooting: DllNotFoundException

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

Resolve this by performing one of these actions:

- Use the NuGet package.
- Add `PdfSplMrgAPI.dll` as an existing item to your project and set its property "Copy to output directory" to "Copy if newer", or

- Add the directory where `PdfSplMrgAPI.dll` resides to the environment variable `%Path%`, or
- Manually copy `PdfSplMrgAPI.dll` to the output directory of your project.

## Troubleshooting: BadImageFormatException

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

(Using the NuGet package normally ensures that the matching native DLL is loaded at execution time.)

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

**AnyCPU** This means that the application runs as a 32-bit process on 32-bit Windows and as 64-bit process on 64-bit Windows. When using AnyCPU, then the correct native DLL must be used, depending on the Windows platform. You can perform this either when installing the application by installing the matching native DLL, or at application start-up by determining the application's platform and ensuring the matching native DLL is loaded. The latter can be achieved by placing both the 32 bit and the 64 bit native DLL in subdirectories `win-x86` and `win-x64` of the application's directory, respectively.

**x86** This means that the application always runs as 32-bit process, regardless of the platform of the Windows installation. The 32-bit DLL runs on all systems.

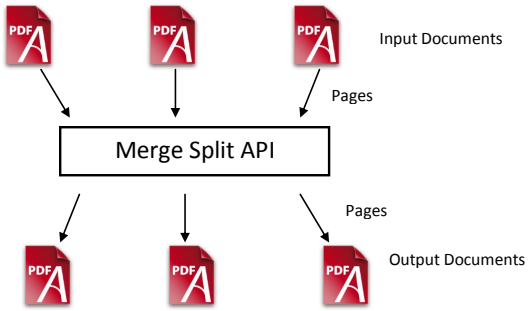
**x64** This means that the application always runs as 64-bit process. As a consequence, the application will not run on a 32-bit Windows system.

# 5 User 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.

> **Note:** Merging PDF 1.x and PDF 2.0 is currently not supported and results in an error code `PDF_E_INVCOMPLIANCE`.

## 5.2 Creating documents that conform to PDF/A

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 conforming documents if the input documents conform to PDF/A 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, the output intent must be 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. 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 conformance 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 conformance, the `ePdfCopyAssociatedFiles` flag must be set (see `TPdfCopyOption`) to copy the embedded file's associations in `CopyPages2`.

## 5.3 Using the API efficiently

### 5.3.1 Copying pages

The Merge Split API supports copying single pages and page ranges. Prior to copying pages, you must define copy options (`TPdfCopyOption`) that specify what to copy and whether to copy 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 `pdfcopydecl.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()
```

**Efficient 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()
```

**Inefficient 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 because when copying a page you need to copy both the page content and the 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 to free associated resources.

### 5.3.3 Features and impact on performance

The performance of the 3-Heights® PDF Merge Split API is determined by the complexity of the input documents and 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, the 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. The return values to be interpreted as failures are documented in the Interface reference. To identify the error on a programmatic level, check the `ErrorCode` property. The `ErrorMessage` property provides a human readable error message, which describes 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 Interface reference

**Note:** This manual describes the COM interface only. Other interfaces (C, Java, .NET) 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.

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

### 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 Error handling). See also enumeration `TPDFErrorCode`. Pdftools 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. This value should only be read if a function call on the PDF Merge Split API has returned a value, which signals a failure of the function (see Error handling)

> **Note:** Reading this property if no error has occurred can yield `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 Page property. 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 Rotate property.

### 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 ErrorCode and ErrorMessage properties 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 OpenStream

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

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

### Parameters:

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

`Password` [`String`] (optional) The user or the owner password of the encrypted PDF document. If this parameter is left out, an empty string is used as a default.

### Returns:

`True` The document could be successfully opened.

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

## 6.1.12 Page

**Property (get, set):** `Long Page`

This property sets and gets 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.

## 6.1.13 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`.

### 6.1.14 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 conforming 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 Mode parameter. 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 CopyOutlineItems2 method.

#### 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 AutoLinearize

**Property (get, set):** `Boolean AutoLinearize`
Default: `False`

> **Note:** With this option enabled, non-Latin characters in the output file name are not supported.

Automatically decide whether to linearize the PDF output file for fast web access.

Applying linearization can lead to a large increase in file size for certain documents. Enabling this option lets the 3-Heights® PDF Merge Split API automatically apply linearization or refrain from doing so based on the estimated file size increase.

With this option enabled, PDF 2.0 documents are automatically excluded from linearization.

See also `Linearize` for more information for linearized PDFs.

> **Note:** If this property is set to `True`, then the value given to `Linearize` is ignored.

### 6.2.7 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.

### 6.2.8 CopyAssociatedFiles

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

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

### 6.2.9 CopyAttributes

**Method:** `Boolean CopyAttributes(InDoc InDoc)`

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

### 6.2.10 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`.)

### 6.2.11 CopyForms

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

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

### 6.2.12 CopyLogicalStructure

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

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

### 6.2.13 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 `Author`, `Keywords`, `Subject`, and `Title` properties).

### 6.2.14 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 `Text` parameter. 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`.)

### 6.2.15 CopyOutlines

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

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

### 6.2.16 CopyOutlineItems

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

Deprecated in Version 4.7. Use `CopyOutlineItems2`.

### 6.2.17 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 `CopyOutlineItems2` method 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
```

## 6.2.18 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.

## 6.2.19 CopyPages

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

Deprecated in Version 4.7. Use CopyPages2 instead.

### 6.2.20 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 also copied. (See `TPdfCopyOption`.)

### 6.2.21 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).

### 6.2.22 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`, and `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 "". 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". 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.

## 6.2.23 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 GetPdf.) For a description of the parameters of CreateInMemory2, see Create.

### Returns:

**True** The PDF document was created successfully.

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

## 6.2.24 CreateAsStream

**Method:** Boolean CreateAsStream(Variant Stream, String UserPw, String OwnerPw)

### Parameter:

**Stream** [Variant] The stream the output file is written to. The stream must support read, write, and random access.

All other parameters and the return value are identical to the Create method.

## 6.2.25 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 Error handling). See also enumeration TPDFErrorCode. Pdftools 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.

## 6.2.26 ErrorMessage

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

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

## 6.2.27 FlattenAnnotations

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

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

## 6.2.28 FlattenFormFields

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

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

## 6.2.29 FlattenSigAppearance

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

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

## 6.2.30 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.

## 6.2.31 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'"
```

## 6.2.32 Keywords

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

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

## 6.2.33 LicenseIsValid

**Property (get):** Boolean LicenseIsValid

Check if the license is valid.

## 6.2.34 Linearize

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

**Note:** This property is ignored when AutoLinearize is set to True.

**Note:** With this option enabled, non-Latin characters in the output file name are not supported.

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

The 3-Heights® PDF Merge Split API does not support linearization of PDF 2.0 documents. For such documents, processing fails. To automatically disable linearization for PDF 2.0, use AutoLinearize.

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 are stored in object streams in the output PDF. For certain input documents this can lead to a significant increase of file size.

## 6.2.35 MergeOptionalContent

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

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

## 6.2.36 OptimizeResources

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

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

## 6.2.37 OutputIntent

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

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

## 6.2.38 PageLayout

**Property (set):** `TPDFPageLayout PageLayout`

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

## 6.2.39 PageMode

**Property (set):** `TPDFPageMode PageMode`

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

### 6.2.40 ProductVersion

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

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

### 6.2.41 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.

### 6.2.42 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`.

### 6.2.43 SetLicenseKey

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

Sets the license key.

### 6.2.44 SetOpenAction

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

The open action defines which page is 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.

## 6.2.45 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"`

## 6.2.46 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 “Author”, “Keywords”, “Subject”, and “Title” properties).

## 6.2.47 SetXMPMetadataMem

**Method:** `Boolean SetXMPMetadataMem(Variant Mem)`

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

## 6.2.48 Subject

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

This property sets the Subject attribute of the document.

### 6.2.49 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), PDF as prefix (.NET) or no prefix at all (Java).

### 6.3.1 TPdfCopyOption Enumeration

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 ePdfCopyLogicalStructure.

**TPDFCopyOption**

| TPDFCopyOption | Description |
|---|---|
| ePdfCopyAnnotations | Copy interactive annotations such as sticky notes or highlight annotations. |
| ePdfCopyFormFields | Copy interactive form fields.<br>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). |

| TPDFCopyOption | |
|---|---|
| ePdfCopyLogicalStructure | Copy logical structure information.<br><br>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.<br><br>It is recommended to use this option if all input documents are "tagged". Otherwise, this could be deactivated to create smaller output files and get a much better performance. This option is required for PDF/A level A conformance (e.g. PDF/A-1a, PDF/A-2a, PDF/A-3a). |
| ePdfCopyNamedDestinations | Copy named destinations.<br><br>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.<br><br>Links within the document 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.<br><br>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 work after merging if ePdfCopyNamedDestinations is used. |
| ePdfCopyOutlines | Copy all outline items (bookmarks) that point to the copied pages.<br><br>The structure of the outline tree in the output document are 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.<br><br>When using this option, it is recommended to leave the ePdfCopyAnnotations flag cleared.<br><br>Note that this option does not flatten form fields, signature appearances, and links, even though technically these are annotations as well. |

## TPDFCopyOption

| | |
|---|---|
| ePdfFlattenFormFields | Flatten form fields preserves the visual appearance of form fields, but discards all interactive elements.<br><br>When using this option, it is recommended to leave the ePdfCopyFormFields flag cleared.<br><br>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. |
| ePdfFlattenSignatureApprearances | Flatten the visual appearance of signed signature fields.<br><br>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 fails and reports 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.<br><br>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, so 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 conformance, this option must be set and the CopyEmbeddedFiles method must be called. |

**TPDFCopyOption**

| | |
|---|---|
| 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.<br>If this option is not set, then no configuration of optional content groups is copied. In this case, you can use the CopyOptionalContent method to copy such configuration information. |
| ePdfSeparateAcroForms | Keep AcroForm fields from different files separate even if they are identical. This option has only an effect, if ePdfCopyFormFields is used.<br>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.<br>If this option is set, fields are renamed if the output document already contains a field of the same name.<br>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. If two fields are merged, multiple widget annotations (i.e. the appearance of the field on pages) are associated with the same form field and therefore always show the same value. |

## 6.3.2 TPDFDestMode Enumeration

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. |

**TPDFDestMode table**

| | | |
|---|---|---|
| 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. |
| 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, see Chapter 8.2.1 in the PDF Reference 1.7.

### 6.3.3 TPDFErrorCode Enumeration

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

The single letter gives an indication of the severity of the error. These are: Success, Error, Warning, and Information. In general, an error is returned if an operation could not be completed, 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 header file bseerror.h, the javadoc documentation of com.pdftools.NativeLibrary.ERRORCODE, and the .NET documentation of Pdftools.Pdf.PDFErrorCode. Note that only a few are relevant for the 3-Heights® PDF 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. |

**TPDFErrorCode table**

| | |
|---|---|
| PDF_E_XFANEEDSRENDERING | The file contains unrendered XFA form fields, i.e. the file is an XFA and not a PDF file.<br>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_E_INVCOMPLIANCE | When merging PDF documents, the conformances of the input files are incompatible, e.g. PDF 1.7 and PDF 2.0. |
| PDF_SPLMRG_W_DOCSIGNED | Document is signed. |
| 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 Enumeration

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

**TPDFPermission table**

| TPDFPermissionFlag | Description |
|---|---|
| ePermNoEncryption | Do not apply encryption.<br>This enumeration value cannot be combined with other values. 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 |

**TPDFPermission table**

| | |
|---|---|
| ePermFillForms | Filling of form fields |
| ePermSupportDisabilities | Support for disabilities |
| ePermAssemble | Document assembly |
| ePermDigitalPrint | High resolution printing |
| ePermAll | Grant all permissions |

Changing permissions or combining multiple permissions is done using a bitwise "or" operator.

> **Note:** The special value ePermNoEncryption cannot be combined with any other values.

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 Enumeration

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, see Chapter 3.6.1 in the PDF Reference 1.7.

### 6.3.6 TPDFPageMode Enumeration

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 | Document outline and thumbnail images are not displayed. |
| ePageModeUseOutlines | The document outline is visible. |
| 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, 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. For example, [C, Java] applies to the C and the Java interface.

## 8.1 Changes in versions 6.19–6.27

- **Update** license agreement to version 2.9

## 8.2 Changes in versions 6.13–6.18

- [.NET, C, Java] **New** methods `OpenStream` and `SaveAsStream` to facilitate the use of streams as input and output.

## 8.3 Changes in versions 6.1–6.12

- [Java] **Changed** minimal supported Java language version to 7 [previously 6].
- [PHP] **Removed** all versions of the PHP interface.
- [.NET] **New** availability of this product as NuGet package for Windows, macOS and Linux.
- [.NET] **New** support for .NET Core versions 1.0 and higher. The support is restricted to a subset of the operating systems supported by .NET Core, see Operating systems.
- [.NET] **Changed** platform support for NuGet packages: The platform "AnyCPU" is now supported for .NET Framework projects.
- **Changed** handling of conformance in all methods that copy from an input document to an output document: The conformance of the target document is adapted automatically and if that's not possible, an error is generated.

## 8.4 Changes in version 5

- **New** additional supported operating system: Windows Server 2019.
- [PHP] **New** extension PHP 7.3 (non thread safe) for Linux.

## 8.5 Changes in version 4.12

- **New** support for encryption according to PDF 2.0 (revision 6, replaces deprecated revision 5).
- **New** HTTP proxy setting in the GUI license manager.
- [.NET, C, COM, Java, PHP] **New** property `AutoLinearize` to automatically choose whether to linearize the output document or not.
- **Changed** values of the error codes `PDF_E_RICHTEXT` and `PDF_W_RICHTEXT`.

## 8.6 Changes in version 4.11

- **New** support for the creation of appearance streams for free text annotations that contain rich text content.
- **New** support for reading and writing PDF 2.0 documents.

- **New** support for the creation of output files larger than 10GB (not PDF/A-1).
- **New** optimization of output file size for documents that contain structure information.

## 8.7 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.
- **New** support for writing PDF objects into object streams. 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.
- **Improved** robustness against corrupt input PDF documents.
- [C] **Clarified** Error handling of `TPdfStreamDescriptor` functions.

## 8.8 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`[6].
- [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.
- [C] **Changed** 32-bit binaries on Windows that link to the API need to be recompiled due to a change of the used mangling scheme.

## 8.9 Changes in version 4.8

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

### Interface `OutDoc`

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

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

# 9 Licensing, copyright, and contact

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

Pdftools 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.



### Contact

PDF Tools AG
Brown-Boveri-Strasse 5
8050 Zürich
Switzerland
https://www.pdf-tools.com
pdfsales@pdf-tools.com