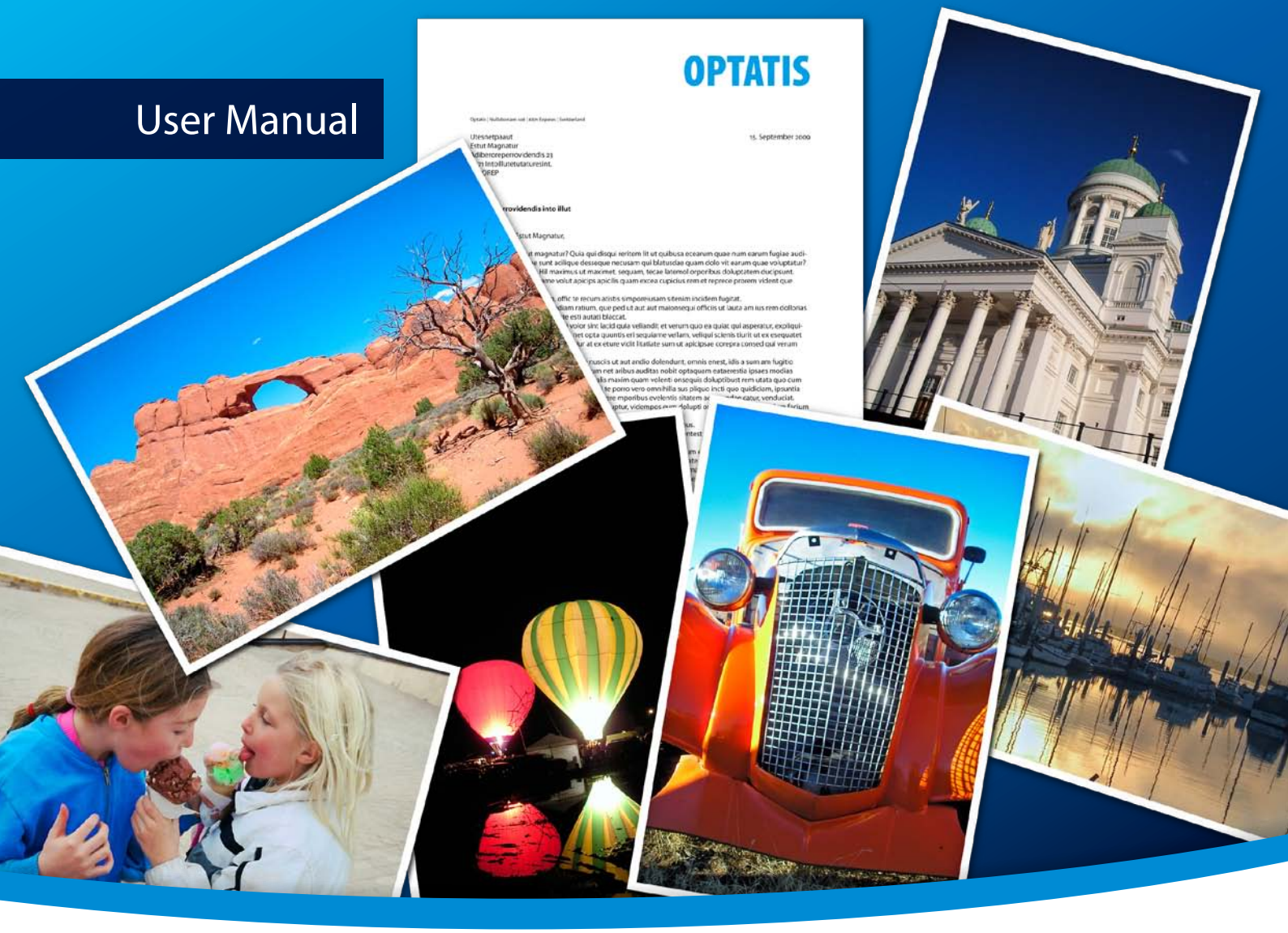
User Manual



3-Heights®
TIFF Toolbox API

PDF/A conform

Version 6.27.8

PDF-TOOLS.COM
Premium PDF Technology

Contents

1 Introduction 5
1.1 Description 5
1.2 Functions 5
1.2.1 Features 5
1.2.2 Formats 6
1.2.3 Conformance 6
1.3 Interfaces 6
1.4 Operating systems 6

2 Installation and deployment 8
2.1 Windows 8
2.2 ZIP archive 8
2.2.1 Development 9
2.2.2 Deployment 10
2.3 NuGet package 11
2.4 Interface-specific installation steps 11
2.4.1 Java interface 11
2.4.2 .NET interface 12
2.4.3 C interface 12
2.5 Uninstall, Install a new version 12
2.6 Note about the evaluation license 12

3 License management 13

4 Getting started 14
4.1 Mixed Raster Content (MRC) 14
4.1.1 Phase 1: Recognizing photographic pictures 14
4.1.2 Phase 2: Segmentation into layers 14
4.1.3 Phase 3: PDF construction 15
4.2 Internal engine 15
4.2.1 `RecognizeBlankPages` 15
4.2.2 `BlankPageMargin` 15
4.2.3 `DisableMaskEmbedding` 16
4.2.4 `RecognizePictures` 16
4.3 Garbage collection and closing objects 16

5 Programming interfaces 17
5.1 .NET interface 17
5.1.1 `IDisposable` Objects 17
5.1.2 Error handling 17
5.1.3 Streams 17
5.1.4 Lists 17
5.2 Java interface 17
5.2.1 `AutoCloseable` Objects 17
5.2.2 Properties 18
5.2.3 Error handling 18
5.2.4 Streams 18
5.2.5 Lists 18
5.3 C interface 18

5.3.1 Namespaces, classes, and methods ... 18
5.3.2 Library initialization ... 19
5.3.3 Objects ... 19
5.3.4 Properties ... 19
5.3.5 Error handling ... 19
5.3.6 Strings ... 19
String return values ... 20
5.3.7 Streams ... 20
5.3.8 Lists ... 20
`List` Interface ... 20
`Count` ... 20
`Get` ... 21
`Append` ... 21

6 Interface reference ... 22
6.1 Common methods ... 22
6.1.1 `CheckLicense` ... 22
6.1.2 `Close` ... 22
6.1.3 `LicenseKey` ... 23
6.1.4 `ProductVersion` ... 23
6.2 `Converter` Interface ... 23
6.2.1 `CreatePdf` ... 23
6.2.2 `AddTiff` ... 24
6.2.3 `SetXmp` ... 24
6.2.4 `SetOutputIntent` ... 25
6.2.5 `SetDefaultColorSpace` ... 25
6.2.6 `ClosePDF` ... 26
6.3 `Compressor` Interface ... 26
6.3.1 `OpenTiff` ... 27
6.3.2 `SaveTiff` ... 27
6.3.3 `CloseTiff` ... 28
6.4 `Recognizer` Interface ... 28
6.4.1 `OpenTiff` ... 28
6.4.2 `SaveTiff` ... 29
6.4.3 `OcrEngines` ... 29
6.4.4 `CloseTiff` ... 30
6.5 `Importer` Interface ... 30
6.5.1 `CreateTiff` ... 30
6.5.2 `AddImg` ... 31
6.5.3 `CloseTiff` ... 31
6.6 `Merger` Interface ... 31
6.6.1 `CreateTiff` ... 32
6.6.2 `AddTiff` ... 32
6.6.3 `CloseTiff` ... 33
6.7 `Splitter` Interface ... 33
6.7.1 `OpenTiff` ... 33
6.7.2 `PageCount` ... 34
6.7.3 `SavePages` ... 34
6.7.4 `CloseTiff` ... 34
6.8 `Scanner` Interface ... 34
6.8.1 `SetSource` ... 35
6.8.2 `ShowUI` ... 35

6.8.3 Scan ... 35
6.8.4 Capabilities ... 36
6.8.5 Sources ... 36
6.9 StringList Interface ... 36
6.9.1 Get ... 36
6.9.2 Size ... 37
6.10 Structures ... 37
6.10.1 ConverterParameters Struct ... 37
Resolution ... 37
OptionalContentGroup ... 37
IgnoreOcr ... 37
Conformance ... 38
6.10.2 CompressorParameters Struct ... 38
BinaryCompression ... 38
GrayscaleCompression ... 38
LabCompression ... 38
PalettedCompression ... 38
Rgb_YcbcrCompression ... 38
CmykCompression ... 38
MrcCompression ... 39
BinaryCompQuality ... 39
GrayscaleCompQuality ... 39
LabCompQuality ... 39
Rgb_YcbcrCompQuality ... 39
CmykCompQuality ... 39
MrcCompQuality ... 40
BinaryDownsampling ... 40
GrayscaleDownsampling ... 40
LabDownsampling ... 40
Rgb_YcbcrDownsampling ... 40
CmykDownsampling ... 40
MrcDownsampling ... 41
Recompress ... 41
PerformMrc ... 41
BinarizationThreshold ... 41
UpgradeJpeg ... 41
6.10.3 RecognizerParameters Struct ... 42
OcrEngineName ... 42
OcrParameters ... 42
OcrLanguages ... 42
BinarizationPriorOcr ... 42
BinarizationThreshold ... 42
6.10.4 ImporterParameters Struct ... 43
Resolution ... 43
UpgradeJpeg ... 43
6.10.5 MergerParameters Struct ... 43
RemoveBlankPages ... 43
6.10.6 SplitterParameters Struct ... 43
PageNumberStart ... 43
PageNumberEnd ... 43
6.11 Enumerations ... 44
6.11.1 Conformance Enumeration ... 44

6.11.2 `BoolEnum` Enumeration 44
6.11.3 `Compression` Enumeration 44
6.11.4 `ErrorCode` Enumeration 45
Logic errors 45
Environmental errors 45

7 Version history 46
7.1 Changes in versions 6.19–6.27 46
7.2 Changes in versions 6.13–6.18 46
7.3 Changes in versions 6.1–6.12 46
7.4 Changes in version 5 46
7.5 Changes in version 4.12 47
7.6 Changes in version 4.11 47
7.7 Changes in version 4.10 47
7.8 Changes in version 4.9 47
7.9 Changes in version 4.8 48

8 Licensing, copyright, and contact 49

1 Introduction

1.1 Description

The 3-Heights® TIFF Toolbox API enables to process and convert TIFF files. Examples of such processing functions are the merging and splitting of multi-page files, the conversion of other raster image formats such as JPEG into TIFF and the conversion of TIFF files into PDF/A. The optional OCR add-in makes output files searchable in full text mode.

PDF/A has been acknowledged world-wide as the ISO standard for long-term archiving since 2005. The TIFF to PDF/A converter is used to convert images into a standardized format, for instance for electronic archiving or electronic data exchange.

The 3-Heights® TIFF Toolbox API is characterized by a robust design, high throughput, ability to process large files and accurate image reproduction.

1.2 Functions

The 3-Heights® TIFF Toolbox API consists of the following interfaces

`Converter` Convert a multi-page TIFF file into a PDF/A-1 or PDF/A-2 file.

`Compressor` Re-compress a multi-page TIFF file using the selected algorithms. Split each page of a multi-page TIFF file with the Mixed Raster Content (MRC) technique into foreground, background and mask layer, compress these layers, sample-down the foreground and background.

`Importer` Import raster image formats such as JPEG into a TIFF file.

`Merger` Merge single- or multi-page TIFF files into one multi-page TIFF file.

`Recognizer` Perform optical character recognition using the optional OCR add-in, calculate mask or detect photographic pictures. Embed the recognized text in the resulting TIFF file.

`Splitter` Split a multi-page TIFF file into single-page TIFF files.

`Scanner` Scan image from a TWAIN or WIA scanner into one multi-page file.

1.2.1 Features

- Support for mixed raster content (MRC) layers during compression and conversion to PDF/A.
- Support for optical character recognition (OCR) and searchable PDFs through optional OCR add-in.
- Support for recognition of photographic picture regions.
- The JPEG import supports the RGB and CMYK color spaces of the JFIF and the Adobe Photoshop file formats.
- Support for "old-JPEG" according to TIFF V6 and "new-JPEG" according to Tech. Note #2 including the conversion from old to new format.
- Compression support for JPEG2000 and JBIG2 in addition to the TIFF V6 algorithms.
- Support for ICC color profiles.
- Support for XMP metadata.
- Substitution of a default resolution where missing.

- Support for merging and splitting of multipage TIFFs.
- Support for scanning with scanners that provide the TWAIN or WIA interface.

1.2.2 Formats

Input formats

- JPEG
- TIFF V6 and Tech. Note #2

Output formats

- TIFF V6 and Tech. Note #2
- PDF 1.x (PDF 1.0, …, PDF 1.7)
- PDF 2.0
- PDF/A-1, PDF/A-2, PDF/A-3

1.2.3 Conformance

Standards

- ISO 32000-1 (PDF 1.7)
- ISO 32000-2 (PDF 2.0)
- ISO 19005-1 (PDF/A-1)
- ISO 19005-2 (PDF/A-2)
- ISO 19005-3 (PDF/A-3)
- TIFF V6 (https://www.adobe.com)
- Tech. Note #2 (http://www.ijg.org)
- RFC 2301 (https://www.ietf.org/)

1.3 Interfaces

The following interfaces are available:

- C
- Java
- .NET Framework
- .NET Core[1]

1.4 Operating systems

The 3-Heights® TIFF Toolbox 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

[1] Limited supported OS versions. Operating systems

'+' indicates the minimum supported version.

2 Installation and deployment

2.1 Windows

The 3-Heights® TIFF Toolbox 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 "TIFF Toolbox 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 ZIP archive

The 3-Heights® TIFF Toolbox API provides three 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: ■ C# ■ VB .NET ■ J# ■ others For a convenient way to use this interface, see NuGet package.
Java	The Java interface.
C	The native C interface is for use with C and C++.

2.2.1 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	C
bin\‹platform›\TiffToolboxAPI.dll	Req.	Req.	Req.
bin*NET.dll	Req.		
bin*NET.xml	Doc.		
bin\‹platform›*.ocr	Opt.	Opt.	Opt.
doc*.pdf	Doc.	Doc.	Doc.
doc\javadoc*.*		Doc.	
include\tifftoolboxapi_c.h			Req.
include*.*			Opt.
jar\TiffToolboxAPI.jar		Req.	
lib\‹platform›\TiffToolboxAPI.lib			Req.
samples*.*	Doc.	Doc.	Doc.

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

File description

Name	Description
bin\‹platform›\TiffToolboxAPI.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.
bin\‹platform›*.ocr	OCR plugin DLLs that are used in combination with the 3-Heights® OCR Enterprise Add-on, which can be purchased as a separate product[2]
doc*.*	Documentation
include*.*	Files to include in your C / C++ project
lib\‹platform›\TiffToolboxAPI.lib	The object file library needs to be linked to the C/C++ project.
jar\TiffToolboxAPI.jar	Java API archive
samples*.*	Sample programs in different programming languages

2.2.2 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 three different interfaces.

Files for deployment

Name	.NET	Java	C
bin\‹platform›\TiffToolboxAPI.dll	Req.	Req.	Req.
bin*NET.dll	Req.		
bin\‹platform›*.ocr	Opt.	Opt.	Opt.
jar\TiffToolboxAPI.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).

[2] These files must reside in the same directory as TiffToolboxAPI.dll.

2.3 NuGet package

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

Installation

The package `PdfTools.TiffToolbox 6.27.8` 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.TiffToolbox 6.27.8`.

Development

The package `PdfTools.TiffToolbox 6.27.8` contains .NET libraries with versions .NET Standard 1.1, .NET Standard 2.0, and .NET Framework 2.0, and native libraries for Windows.

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® TIFF Toolbox 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.

2.4 Interface-specific installation steps

2.4.1 Java interface

The 3-Heights® TIFF Toolbox API requires Java version 7 or higher.

For compilation and execution

When using the Java interface, the Java wrapper `jar\TiffToolboxAPI.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\TiffToolboxAPI.jar" ^
 sampleApplication.java
```

For execution

Additionally, the library `TiffToolboxAPI.dll` needs be in one of the system's library directories[3] 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[4].

```
java -classpath ".;C:\Program Files\PDF Tools AG\TiffToolboxAPI.jar" ^
```

[3] On Windows defined by the environment variable PATH, and 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`"

```
"-Djava.library.path=C:\Program Files\PDF Tools AG\bin\x64" sampleApplication
```

2.4.2 .NET interface

The 3-Heights® TIFF Toolbox 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. `TiffToolboxAPI.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 `TiffToolboxAPI.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 `TiffToolboxAPI.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 `TiffToolboxAPI.dll` are copied to subdirectories `win-x86` and `win-x46` of the output directory, respectively.

2.4.3 C interface

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

2.5 Uninstall, Install a new version

If you have used the ZIP file for the installation, 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. If using the COM interface, the new DLL must be registered, de-registering the old version is not required.

2.6 Note about the evaluation license

With the evaluation license, the 3-Heights® TIFF Toolbox API automatically adds a watermark to the output files.

3 License management

The 3-Heights® TIFF Toolbox 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 Getting started

4.1 Mixed Raster Content (MRC)

Some raster images—typically scanned documents—consist mainly of text, possibly in several colors and interspersed with some pictures. Such images are difficult to compress with one single compression type because of the diverse or even conflicting features of different parts of the image.

The MRC technique is a way of breaking such images down into parts, such that each part is well suited for one type of a compression algorithm. With this approach, the resulting file size often can be reduced without significantly reducing the visual quality of the document.

> **Note:** MRC optimization can only be enabled for continuous images, i.e. not for bi-tonal images and images with an indexed color space.

4.1.1 Phase 1: Recognizing photographic pictures

In this phase, the `Recognizer` computes a bi-tonal mask. Optionally rectangular areas containing photographic features are detected. The mask and the information about the location of the detected photographic pictures is embedded in the resulting TIFF.

It is possible, that actual photographic regions present in the input image are not recognized correctly. This can happen for example if a photographic region contains parts with uniform color.

4.1.2 Phase 2: Segmentation into layers

In this phase the `Compressor` is used.

All photographic pictures that have been recognized in phase 1 are removed from the input image and are embedded as separate images in the resulting TIFF file. Thereafter the image is supposed not to contain photographic features anymore. Instead, the image is assumed to consist of text and graphic, potentially with varying color.

Now, using the mask from phase 1 the whole image is separated into three layers, a foreground, a background and a mask layer. The mask, which can be thought of as a bi-tonal image tells for each pixel whether to show the corresponding pixel of the foreground layer or the background layer.

Example:

Let the image consist of a yellow background with black paragraph text and a title text in red. Then the resulting background layer contains the yellow color only. The foreground layer contains the black text color where the paragraph text is located and the red text color where the title is located. In the mask, pixels for which the foreground layer should be displayed are set to 1, the others are set to 0. I.e. the mask contains 1's where the black and the red text is and 0's everywhere else.

In the resulting TIFF the foreground layer, the background layer and the mask are stored as three images. The mask can be compressed differently to the foreground and background layer. Since all the detailed features have been moved to the mask, it makes sense to downsample the foreground and background layers and use a low image quality. The mask on the other hand is usually stored with a lossless compression type optimized for text.

Note: Depending on the size of the input image it is possible that the algorithm decides that the whole input image consists of one photographic region covering the whole image. In this case, this second phase is omitted.

Note: The isolated pictures from phase 1 are not downsampled nor compressed.

4.1.3 Phase 3: PDF construction

In this phase the TIFF-file that resulted from phase 1 (recognition of photographic pictures and recognition of mask) and phase 2 (the segmentation into foreground and background layers and into a mask) is now used to construct a PDF-file by using the `Converter`. If in phase 1, a single photographic region covering the entire image is detected, then the original image is used and the reconstruction is finished. Otherwise, the construction first places the background layer, followed by the foreground layer with the mask. Finally, if any isolated photographic pictures are found they are placed at their respective locations on top of the foreground layer.

4.2 Internal engine

The internal engine comprises recognition functionality for the `Recognizer`, such as mask recognition or recognition of photographic picture regions.

In general, the string parameter for the OCR engines is composed by a sequence of Key-Value pairs that are separated by semicolons (;). In order to form the string parameter the following keys are supported by the internal engine.

4.2.1 RecognizeBlankPages

Key: `RecognizeBlankPages` Type: `Boolean` Default: `false`

Recognize blank pages of a certain file. A blank page is considered to be a page with a uniform coloring containing only slight noise. Colored, grayscale and bi-tonal pages can be subject to blank page recognition. The value of the Key-Value pair takes either `true` or `false`.

Example: Choose internal engine and recognize blank pages. Set parameters in `RecognizerParameters` as follows:

```
ocrEngineName = "internal";
ocrParameters = "RecognizeBlankPages=true";
```

Blank pages can then be removed from a tiff file using the `Merger`.

4.2.2 BlankPageMargin

Key: `BlankPageMargin` Type: `double` Default: `0.02`

Set the ratio the margin takes with respect to the corresponding page length. The margin is excluded from the analysis whether a page is blank. The allowed values range from 0 to 0.5. This parameter is only active if at the same time the value of `RecognizeBlankPages` is `true`.

Example: Choose internal engine and analyze if a page is blank without taking into account a margin of 5% on every side. Set parameters in `RecognizerParameters` as follows:

```
ocrEngineName = "internal";
ocrParameters = "RecognizeBlankPages=true;BlankPageMargin=0.05";
```

Blank pages can then be removed from a tiff file using the `Merger`.

4.2.3 DisableMaskEmbedding

Key: `DisableMaskEmbedding` Type: `Boolean` Default: `false`

If this option is set to `true`, no mask is embedded in the output TIFF. If this option is not set, a mask is embedded by default. The value of the Key-Value pair takes either `true` or `false`.

Example: Choose internal engine, recognize photographic pictures, but don't embed mask in output. Set parameters in `RecognizerParameters` as follows:

```
ocrEngineName = "internal";
ocrParameters = "DisableMaskEmbedding=true;RecognizePictures=true";
```

4.2.4 RecognizePictures

Key: `RecognizePictures` Type: `Boolean` Default: `false`

Recognize photographic picture regions.

Example: Choose internal engine and recognize photographic pictures. Set parameters in `RecognizerParameters` as follows:

```
ocrEngineName = "internal";
ocrParameters = "RecognizePictures=true";
```

The recognition of photographic picture regions works only for colored pictures.

4.3 Garbage collection and closing objects

Every interface object is considered being a resource that needs to be closed after use. Most objects are closed automatically, at the latest when the owning document is closed, in C# and Java possibly earlier by the garbage collector.

5 Programming interfaces

Where possible and useful, the 3-Heights® TIFF Toolbox API uses language specific features. This means that some parts of the API use different syntax for different programming languages.

5.1 .NET interface

5.1.1 `IDisposable` Objects

Objects that must be closed explicitly implement the `IDisposable` interface. Instead of calling `Dispose()` directly, it is recommended that you use the "`using`" statement:

```
using (Document document = ...)
{
  ...
} // document.Dispose() is called implicitly here
```

See also Garbage collection and closing objects.

5.1.2 Error handling

Errors are reported using exceptions.

The three logic error codes are mapped to the corresponding native exception classes:

`IllegalArgument` maps to `System.ArgumentException`

`IllegalState` maps to `System.InvalidOperationException`

`UnsupportedOperation` maps to `System.NotSupportedException`

The rest of the error codes is modeled using a single exception class `PdfTools.ErrorCodeException` that provides access to the underlying error code and message.

5.1.3 Streams

The native stream interface `System.IO.Stream` is used.

5.1.4 Lists

Lists implement the native list interface `System.Collections.Generic.IList<T>`.

5.2 Java interface

5.2.1 `AutoCloseable` Objects

Objects that must be closed explicitly implement the `AutoCloseable` interface. Instead of calling `close()` directly, it is recommended that you use the "try-with-resources" statement:

```
try (Document document = ...) {
  ...
} // document.close() is called implicitly here
```

See also Garbage collection and closing objects.

5.2.2 Properties

Properties are modeled with setter and getter methods.

5.2.3 Error handling

Errors are reported using exceptions.

The three logic error codes are mapped to the corresponding native runtime exception classes and are not checked:

`IllegalArgument` maps to `java.lang.IllegalArgumentException`

`IllegalState` maps to `java.lang.IllegalStateException`

`UnsupportedOperation` maps to `java.lang.UnsupportedOperationException`

The rest of the error codes is modeled using a single checked exception class `com.pdf_tools.ErrorCodeException` that provides access to the underlying error code and message.

5.2.4 Streams

The native stream interfaces cannot be used, because they are lacking two important features:

- The PDF file format is based on random access. Native Java streams have only limited support for this.
- The ability to read from an output stream is crucial for processing large files.

Instead, a custom stream interface `com.pdf_tools.Stream` is provided, which has a similar interface as `java.io.RandomAccessFile`.

An implementation for files is provided, backed by `java.io.RandomAccessFile`.

5.2.5 Lists

Lists implement the native Java list interface `java.util.List`.

5.3 C interface

5.3.1 Namespaces, classes, and methods

In most languages, namespaces and classes are used to model the interfaces.

The exception is C, where this is modeled with function prefixes and functions operating on handles. The prefix of all functions of the 3-Heights® TIFF Toolbox API is `Tiff`, for types, it is `TTiff` and for enum values, `eTiff`.

5.3.2 Library initialization

The first method called must be `TiffInitialize`. Failing to invoke this function results in undefined behavior. Similarly, the last method must be `TiffUninitialize`.

5.3.3 Objects

Objects in the C interface are represented by object handles. After use, all object handles returned by the 3-Heights® TIFF Toolbox API must be closed with `TiffClose`.

5.3.4 Properties

Properties are modeled with setter and getter methods.

5.3.5 Error handling

After a having called a method, an error should be detected as follows:

a. If the method's return type is `BOOL` or a pointer, and the return value is `FALSE` or `NULL`, respectively, then an error has occurred.
b. If the method's return type is other than `BOOL` or a pointer, then `TiffGetLastError` must be called to detect whether an error has occurred.

More information about the error can be retrieved by using the functions `TiffGetLastError` and `TiffGetLastErrorMessage`.

5.3.6 Strings

All functions involving strings are provided in two different flavors:

- UTF-16 function with suffix `W`, using `WCHAR` as parameter type.
- Multibyte character set function with suffix `A`, using `char` as parameter type. The concrete character set that is used depends on the platform:
  - On Windows, the current ANSI code page (`CP_ACP`) is assumed.
  - On Linux or macOS, the current C encoding (`LC_CTYPE`) is used.

In addition to the effective function names with suffix, there's a macro without suffix for each function pair: It either resolves to the `W` variant (if `_UNICODE` is defined), or to the `A` variant (if `_UNICODE` is **not** defined).

Example: Signature of an API string property setter, where `‹String›` stands for the property's name:

```
void TiffSet‹String›A(const char* szString);  // Multibyte encoding
void TiffSet‹String›W(const WCHAR* szString); // UTF-16
#ifdef _UNICODE
#define TiffSet‹String› TiffSet‹String›W
#else
#define TiffSet‹String› TiffSet‹String›A
#endif
```

String return values

Functions that return a string are treated specially in C. Instead of returning the string, those functions take a buffer and size as last parameters and write into that buffer. The return value is the amount of data written to the buffer.

To determine the required buffer size, the function has to be called with `NULL` as argument.

Calling the function with a buffer size that is too small results in a `ePdfErrorIllegalState`.

Multibyte character set functions (with suffix `A`) that return a string can fail to encode the string in the current operating systems' encoding. In case of such a failure, the return value is `0` and no error code is set. To prevent such failures, it is recommended that you use the UTF-16 (`W`) functions on Windows or to use operating systems with a Unicode code page.

Example: Signature and usage of an API string property getter (error handling is omitted), where `‹String›` stands for the property's name:

```
size_t TiffGet‹String›A(char* pBuffer, size_t nBufferSize);
```

```
size_t nBufferSize = TiffGet‹String›A(NULL, 0);
char* pBuffer = malloc(nBufferSize * sizeof char);
nBufferSize = TiffGet‹String›A(pBuffer, nBufferSize);
```

5.3.7 Streams

Streams are modeled by means of a set of callbacks and a context pointer, grouped in a struct `TPdfStreamDescriptor`.

An implementation for `FILE*` is provided in the header file `pdfdecl.h`. (Search for function `PdfCreateFILEStreamDescriptor`.)

5.3.8 Lists

Lists in C are implemented like any other interface.

`List` Interface

Every list type provides a subset of the following properties and methods, where `‹ElementType›` stands for the type name of the contained elements:

`Count`

Property (get): `int Count`

The number of elements of the list.

Get

Method: `‹ElementType› Get(int index)`
Error codes: `IllegalState`, `IllegalArgument`, `UnsupportedOperation`

Get an element of the list.

Append

Method: `Append(‹ElementType› element)`
Error codes: `IllegalState`, `IllegalArgument`, `UnsupportedOperation`

Append an element to the list.

6 Interface reference

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

6.1 Common methods

Note: In this section common static methods are listed. They can be called in every interface.

6.1.1 CheckLicense

Method: `void CheckLicense()`
Static
Error code: `License`

Check if the product is properly licensed.

This method can be used to perform a license check without actually opening or creating a document, e.g. when starting a GUI application or a service.

Error Code:

`License` The product is not properly licensed.

6.1.2 Close

Method: `void Close()`
Error code: `IllegalState`

Close the object and release all native resources associated with it.

After closing the object and releasing all native recources, any call to a method of this object will result in an `IllegalState` error.

Error Code:

`IllegalState` The object has already been closed.

6.1.3 LicenseKey

Property (set): `String LicenseKey`
Static
Error code (set): `License`

Set the license key.

> **Note:** License keys that require activation can only be installed in the license manager. Setting them at runtime is not supported.

Error Code:

`License` The license key is not valid.

6.1.4 ProductVersion

Property (get): `String ProductVersion`
Static

Get the version of the 3-Heights® TIFF Toolbox API in the format "A.C.D.E".

6.2 Converter Interface

The Converter Interface converts a stream of a multi-page TIFF file into a stream of a PDF/A-2 conforming document.

If a TIFF page contains mixed raster content (MRC) layers (background, foreground, mask) they are converted into equivalent objects in the PDF document. The same is true for TIFF pages which contain alpha channels.

If a TIFF page contains embedded OCR text it is converted into an invisible text layer to make the resulting PDF document searchable. Note that only embedded OCR text is supported that was created using the Converter Interface. Tools which create embedded OCR text from other vendors are not supported, since the embedding of OCR is not part of the TIFF V6 standard and all vendors are using proprietary tags.

If a TIFF page contains XMP metadata stream it is added as a Metadata entry in the PDF page object.

In addition to the device specific color spaces the Converter Interface also supports calibrated color spaces and ICC color profiles.

The Converter Interface minimizes the re-compression of streams. If the page is not compressed it will not be compressed in the resulting output. However, if the page uses LZW compression it will be re-compressed using FLATE, since LZW is not allowed in PDF/A-2.

6.2.1 CreatePdf

Method: `void CreatePdf(Stream stream, ConverterParameters convParms)`
Error codes: `IO`, `License`, `IllegalArgument`, `IllegalState`, `UnsupportedOperation`

Create the output PDF file.

Parameters:

stream [Stream] The stream of the output PDF file.

convParms [ConverterParameters] Parameter object for the conversion.

Error Codes:

IO Error occurred writing to the stream.

License License not valid.

IllegalArgument The stream argument is null.

IllegalState The output document has already been created.

UnsupportedOperation Creation of output document failed.

6.2.2 AddTiff

Method: void AddTiff(Stream stream)
Error codes: IO, UnsupportedOperation, IllegalArgument, IllegalState

Add TIFF file to PDF output file.

Parameter:

stream [Stream] The stream of the input TIFF file.

Error Codes:

IO Error occurred reading from the stream.

UnsupportedOperation Adding of input TIFF document failed

IllegalArgument The stream argument is null.

IllegalState The output document has already been closed.

6.2.3 SetXmp

Method: void SetXmp(Stream stream)
Error codes: IO, IllegalState, IllegalArgument, UnsupportedOperation

Set XMP metadata.

Parameter:

stream [Stream] The stream of the XMP metadata.

Error Codes:

IO Error occurred reading from the stream.

IllegalState The output document has not been created yet.

IllegalArgument The stream argument is null.

UnsupportedOperation Setting XMP metadata failed.

6.2.4 SetOutputIntent

Method: Boolean SetOutputIntent(Stream stream)
Error codes: IO, IllegalState, UnsupportedOperation

The output intent represents the output color profile. Setting the output intent is generally only recommended for images intended for a particular CMYK output device. In other cases, the SetDefaultColorSpace method should be used.

Color profiles are usually provided with the operating system. On Windows, for example, they can be found at C:\Windows\System32\spool\drivers\color. Alternatively, profiles can be found here:

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

Most color profiles are copyrighted. Therefore, you should read the license agreements on the above links before using the color profiles.

Parameter:

stream [Stream] The stream of the color profile. An example could be:

C:\Windows\System32\spool\drivers\color\USWebCoatedSWOP.icc

If PDF/A conformance is selected and no output intent is defined, then CMYK USWebCoatedSWOP.icc is embedded as default output intent.

This method must be called after CreatePdf has been called.

6.2.5 SetDefaultColorSpace

Method: void SetDefaultColorSpace(Stream stream)
Error codes: IO, IllegalState, IllegalArgument, UnsupportedOperation

This color space is used in the output PDF for pages that have no ICC profile available in the corresponding page in the input TIFF. A default color space profile can be set for both RGB and CMYK.

Parameter:

`stream` `[Stream]` The stream to the color space profile.

Error Codes:

`IO` Error occurred reading from the stream.

`IllegalState` The output document has not been created yet.

`IllegalArgument` The `stream` argument is `null`.

`UnsupportedOperation` Setting default color space failed.

6.2.6 ClosePDF

Method: `void ClosePDF()`
Error code: `IllegalState`

Close output PDF file. In comparision to the `Close` method the object itself is not closed and hence can be used for further processing.

Error Code:

`IllegalState` The output document has already been closed.

6.3 Compressor Interface

The Compressor Interface compresses the pages of the input file with the selected compression algorithms.

For each of the TIFF classes Binary, Grayscale, Lab, Palette, RGB & YCbCr, Separated (CMYK) and MRC a different compression algorithm can be specified. The tool also allows for specifying a class specific quality measure for lossy compression algorithms.

The Compressor Interface allows for upgrading embedded JPEG streams in the TIFF V6 format to the newer format specified in the independent JPEG group's Technical Note #2.

The Compressor Interface provides the possibility of using the mixed raster content (MRC) technique, i.e. continuous images can be converted into background, foreground and mask layers (Section 4.1). The computation of a mask is done in the `Recognizer`. If no mask is present, the Compressor interface computes one when performing MRC. Photographic picture regions that are recognized by the `Recognizer` are not subject to MRC. The foreground and background layers can additionally be downsampled.

The Compressor Interface processes the stream of an input single- or multi-page TIFF file. The output file contains the same number of pages but with differently compressed image data. If MRC layers are being created, the output file contains for every page images for the mask, the foreground and background layer, and the isolated photographic pictures that were recognized by the `Converter`.

6.3.1 OpenTiff

Method: `void OpenTiff(Stream stream)`
Error codes: `IO`, `License`, `IllegalArgument`, `IllegalState`

Open the input TIFF file.

Parameter:

`stream` [`Stream`] The stream of the input TIFF file.

Error Codes:

`IO` Error occurred reading from the stream.

`License` License not valid.

`IllegalArgument` The `stream` argument is `null`.

`IllegalState` The input document has already been opened.

6.3.2 SaveTiff

Method: `void SaveTiff(Stream stream, CompressorParameters compParms)`
Error codes: `IO`, `IllegalArgument`, `IllegalState`, `UnsupportedOperation`

Compress and save the TIFF file.

Parameters:

`stream` [`Stream`] The stream of the output TIFF file.

`compParms` [`CompressorParameters`] Parameter object for the compression.

Error Codes:

`IO` Error occurred writing to the stream.

`IllegalArgument` The `stream` argument is `null`.

`IllegalState` The input document has already been closed.

`UnsupportedOperation` Creation of output document failed.

6.3.3 CloseTiff

Method: `void CloseTiff()`
Error code: `IllegalState`

Close input TIFF file. In comparision to the `Close` method the object itself is not closed and hence can be used for further processing.

Error Code:

`IllegalState` The output document has already been closed.

6.4 Recognizer Interface

The Recognizer Interface recognizes the text using an OCR engine and embeds the recognized text. Furthermore a binary mask is calculated and photographic picture regions are detected.

In order to use text recognition one of the supported OCR engines must be used. Currently the engines FineReader 10 and 11 from ABBYY are supported. These engines are available as separate product kits.

Furthermore there is an internal engine which provides some recognition functionality (see Section 4.2), such as recognition of a binary mask or recognition of photographic picture regions.

Mask computation is done by default. An embedded mask can then be used in the `Compressor` to perform MRC (Section 4.1).

The Recognizer Interface processes the stream of one input single- or multi-page TIFF file. The output stream describes a TIFF file containing the same number of pages. If the OCR process for a specific page succeeds then the original image is replaced by the one delivered by the OCR engine, if any. The new image is compressed using a lossless algorithm such as CCITT G4 and LZW depending on the image class.

6.4.1 OpenTiff

Method: `void OpenTiff(Stream stream)`
Error codes: `IO`, `License`, `IllegalArgument`, `IllegalState`

Open the input TIFF file.

Parameter:

`stream` `[Stream]` The stream of the input TIFF file.

Error Codes:

`IO` Error occurred reading from the stream.

`License` License not valid.

`IllegalArgument` The `stream` argument is `null`.

IllegalState The input document has already been opened.

6.4.2 SaveTiff

Method: `void SaveTiff(Stream stream, Stream xmlStream, RecognizerParameters recogParms)`
Error codes: `IO`, `IllegalArgument`, `IllegalState`, `UnsupportedOperation`

Recognize text and / or photographic picture regions and save resulting TIFF file.

Parameters:

stream [Stream] The stream of the output TIFF file.

xmlStream [Stream] The stream of the XML file that stores the information of the recognized text and / or photographic image regions. If this stream is `null` no information of the recognition is returned.

recogParms [RecognizerParameters] Parameter object for the recognition of text and / or photographic image regions.

Error Codes:

IO

- Error writing to the stream.
- Error writing to the XML stream.

IllegalArgument

- The `stream` argument is `null`.
- The `recogParms.ocrParameters` argument has a value that is is not supported.
- The `recogParms.ocrLanguages` argument has a value that is is not supported.
- The `recogParms.ocrEngineName` argument has a value that is is not supported.

IllegalState The input document has already been closed.

UnsupportedOperation Creation of output document failed.

6.4.3 OcrEngines

Property (get): `StringList OcrEngines`

Get list of OCR engines.

6.4.4 CloseTiff

Method: `void CloseTiff()`
Error code: `IllegalState`

Close input TIFF file. In comparision to the `Close` method the object itself is not closed and hence can be used for further processing.

Error Code:

`IllegalState` The output document has already been closed.

6.5 Importer Interface

The Importer Interface imports a stream of a JPEG, TIFF or PDF image and embeds it into a stream of a TIFF container.

The JPEG streams are embedded conforming to the TIFF V6 specification. The Interface supports JPEG streams conforming to the JFIF and Adobe Photoshop format by interpreting the corresponding application specific markers.

If the input stream contains ICC profiles or XMP metadata streams they are embedded separately using the appropriate tags. All other application specific markers are ignored.

If the resolution information is missing in the JPEG stream a default resolution is used. The color space is retrieved from the application specific markers and, if missing, from the number of color channels.

6.5.1 CreateTiff

Method: `void CreateTiff(Stream stream, ImporterParameters importParms)`
Error codes: `IO`, `License`, `IllegalArgument`, `IllegalState`, `UnsupportedOperation`

Create the output TIFF file.

Parameters:

`stream` `[Stream]` The stream of the output TIFF file.

`importParms` `[ImporterParameters]` Parameter object for the conversion.

Error Codes:

`IO` Error occurred writing to the stream.

`License` License not valid.

`IllegalArgument` The `stream` argument is `null`.

`IllegalState` The output document has already been created.

`UnsupportedOperation` Creation of output document failed.

6.5.2 AddImg

Method: `void AddImg(Stream stream)`
Error codes: `IO`, `UnsupportedOperation`, `IllegalArgument`, `IllegalState`

Add Image (JPEG, TIFF or PDF) to TIFF output file.

Parameter:

`stream` [`Stream`] The stream of the input Image file.

Error Codes:

`IO` Error occurred reading from the stream.

`UnsupportedOperation` Adding of input Image failed.

`IllegalArgument` The `stream` argument is `null`.

`IllegalState` The output document has already been closed.

6.5.3 CloseTiff

Method: `void CloseTiff()`
Error code: `IllegalState`

Close output TIFF file. In comparision to the `Close` method the object itself is not closed and hence can be used for further processing.

Error Code:

`IllegalState` The output document has already been closed.

6.6 Merger Interface

The Merger Interface merges streams of single- and multi-page TIFF files into a stream of a large multi-page file without further processing except for embedded JPEG streams.

There is a special processing of embedded V6 JPEG streams. Instead of just copying the corresponding tags to the output the JPEG stream is re-assembled from the corresponding tags and then embedded into the output page conforming to the TIFF V6 specification. This fixes known problems with badly created TIFF V6 files (see Technical Note #2 for more information on this) and increases the interoperability with the most common readers (viewers) in use.

6.6.1 CreateTiff

Method: `void CreateTiff(Stream stream, MergerParameters mergeParms)`
Error codes: `IO`, `License`, `IllegalArgument`, `IllegalState`, `UnsupportedOperation`

Create the output TIFF file.

Parameters:

`stream` [`Stream`] The stream of the output PDF file.

`mergeParms` [`MergerParameters`] Parameter object that tells whether blank pages should be removed.

Error Codes:

`IO` Error occurred writing to the stream.

`License` License not valid.

`IllegalArgument` The `stream` argument is `null`.

`IllegalState` The output document has already been created.

`UnsupportedOperation` Creation of output document failed.

6.6.2 AddTiff

Method: `void AddTiff(Stream stream)`
Error codes: `IO`, `UnsupportedOperation`, `IllegalArgument`, `IllegalState`

Add TIFF to output TIFF file.

Parameter:

`stream` [`Stream`] The stream of the input TIFF file.

Error Codes:

`IO` Error occurred reading from the stream.

`UnsupportedOperation` Adding of input TIFF document failed

`IllegalArgument` The `stream` argument is `null`.

`IllegalState` The output document has already been closed.

6.6.3 CloseTiff

Method: `void CloseTiff()`
Error code: `IllegalState`

Close output TIFF file. In comparision to the `Close` method the object itself is not closed and hence can be used for further processing.

Error Code:

`IllegalState` The output document has already been closed.

6.7 Splitter Interface

The Splitter Interface selects a certain page from the input stream of a multi-page TIFF file and returns a stream of the corresponding single-page file without further processing except for embedded JPEG streams.

There is a special processing of embedded V6 JPEG streams. Instead of just copying the corresponding tags to the output the JPEG stream is re-assembled from the corresponding tags and then embedded into the output page conforming to the TIFF V6 specification. This fixes known problems with badly created TIFF V6 files (see Technical Note #2 for more information on this) and increases the interoperability with the most common readers (viewers) in use.

6.7.1 OpenTiff

Method: `void OpenTiff(Stream stream)`
Error codes: `IO`, `License`, `IllegalArgument`, `IllegalState`

Open the input TIFF file.

Parameter:

`stream` `[Stream]` The stream of the input TIFF file.

Error Codes:

`IO` Error occurred reading from the stream.

`License` License not valid.

`IllegalArgument` The `stream` argument is `null`.

`IllegalState` The input document has already been opened.

6.7.2 PageCount

Property (get): `int PageCount`

Get list of OCR engines.

6.7.3 SavePages

Method: `void SavePages(Stream stream, SplitterParameters splitParms)`
Error codes: `IO`, `IllegalArgument`, `IllegalState`, `UnsupportedOperation`

Pick all pages in a certain page range of the input TIFF file and save them as new TIFF file.

Parameters:

`stream` [`Stream`] The stream of the output TIFF file.

`splitParms` [`SplitterParameters`] Parameter object that contains page range.

Error Codes:

`IO` Error occurred writing to the stream.

`IllegalArgument` The `stream` argument is `null`.

`IllegalState` The input document has already been closed.

`UnsupportedOperation` Creation of output document failed.

6.7.4 CloseTiff

Method: `void CloseTiff()`
Error code: `IllegalState`

Close input TIFF file. In comparision to the `Close` method the object itself is not closed and hence can be used for further processing.

Error Code:

`IllegalState` The output document has already been closed.

6.8 Scanner Interface

This interface allows to scan with scanners that provide the TWAIN or WIA interface. One can specify custom scan settings in a user interface and save them in a capability file.

6.8.1 SetSource

Method: `void SetSource(String source)`
Error codes: `IllegalState`, `IllegalArgument`, `IllegalArgument`

Set TWAIN or WIA scanner to scan with. A list of available scanners can be obtained from `Sources`.

Parameter:

`source` `[String]` Name of the scanner.

Error Codes:

`IllegalState` The scanner has already been closed.

`IllegalArgument` The scanner could not be found.

6.8.2 ShowUI

Method: `Boolean ShowUI()`
Error code: `IllegalState`

Show user interface that allows to configure scan settings. Get the saved settings with property `Capabilities`.

Returns:

`true`: User interface button clicked "OK". `false`: User interface button clicked "Cancel".

Error Code:

`IllegalState` The scanner has already been closed.

6.8.3 Scan

Method: `void Scan(Stream stream)`
Error codes: `IO`, `License`, `IllegalState`, `IllegalArgument`, `UnsupportedOperation`

Scan into output stream. Use `SetSource` to set the TWAIN or WIA scanner.

Parameter:

`stream` `[Stream]` Stream to scan into.

Error Codes:

IO Error writing to the `stream`.

License License not valid.

IllegalState The scanner has already been closed.

IllegalArgument The `stream` is `null`.

UnsupportedOperation The output file couldn't be created.

6.8.4 Capabilities

Property (get, set): `Stream Capabilities`
Error codes (set): `IO`, `IllegalState`, `IllegalArgument`, `UnsupportedOperation`

Get or set capability stream which stores scan settings. They can be customized by `ShowUI` in a user interface.

Error Codes:

IO Error writing the `Capabilities` stream.

IllegalState The scanner has already been closed.

IllegalArgument The `Capabilities` stream is `null`.

UnsupportedOperation The `Capabilities` could not be set.

6.8.5 Sources

Property (get): `StringList Sources`

Get list of available TWAIN or WIA sources. Only scanners that match the architecture of the API are listed.

6.9 StringList Interface

6.9.1 Get

Method: `String Get(int index)`
Error code: `IllegalArgument`

Pick certain element of list.

Parameter:

`index` `[int]` Index of list element to be picked.

Error Code:

`IllegalArgument` The `index` is out of bounds.

6.9.2 Size

Method: `int Size()`

Get number of listed elements.

6.10 Structures

6.10.1 ConverterParameters Struct

Resolution

Property (get, set): `double Resolution`
Default: `96.0`

If a TIFF page does not contain resolution tags they are defined by the parameter value of this property. The value's unit is dots per inch (DPI).

OptionalContentGroup

Property (get, set): `BoolEnum OptionalContentGroup`
Default: `False`

PDF/A-2 supports optional content sometimes referred to as layers. If this property is used in conjunction with MRC layers, then the output document contains the optional content groups "Colored Text", "Black Text", "Foreground" and "Background" which can be individually turned on and off.

IgnoreOcr

Property (get, set): `BoolEnum IgnoreOcr`
Default: `False`

Ignore OCR data that was recognized by the `Recognizer`. Photographic image regions are not ignored by this option.

Conformance

Property (get, set): `Conformance Conformance`
Default: `PDF/A-2b`

PDF Conformance level. See `Conformance`.

6.10.2 CompressorParameters Struct

BinaryCompression

Property (get, set): `TiffCompression BinaryCompression`

Compression algorithm for binary images.

GrayscaleCompression

Property (get, set): `TiffCompression GrayscaleCompression`

Compression algorithm for grayscale images.

LabCompression

Property (get, set): `TiffCompression LabCompression`

Compression algorithm for lab images.

PalettedCompression

Property (get, set): `TiffCompression PalettedCompression`

Compression algorithm for paletted images.

Rgb_YcbcrCompression

Property (get, set): `TiffCompression Rgb_YcbcrCompression`

Compression algorithm for RGB and YCbCr images.

CmykCompression

Property (get, set): `TiffCompression CmykCompression`

Compression algorithm for CMYK images.

MrcCompression

Property (get, set): `TiffCompression MrcCompression`

Compression algorithm for MRC images.

BinaryCompQuality

Property (get, set): `double BinaryCompQuality`
Default: `80`

Compression quality for binary images. The value range is from `1` to `100`.

GrayscaleCompQuality

Property (get, set): `double GrayscaleCompQuality`
Default: `80`

Compression quality for grayscale images. The value range is from `1` to `100`. The quality is effective only when using a lossy compression algorithm for the specific class.

LabCompQuality

Property (get, set): `double LabCompQuality`
Default: `80`

Compression quality for lab images. The value range is from `1` to `100`. The quality is effective only when using a lossy compression algorithm for the specific class.

Rgb_YcbcrCompQuality

Property (get, set): `double Rgb_YcbcrCompQuality`
Default: `80`

Compression quality for RGB and YCbCr images. The value range is from `1` to `100`. The quality is effective only when using a lossy compression algorithm for the specific class.

CmykCompQuality

Property (get, set): `double CmykCompQuality`
Default: `80`

Compression quality for CMYK images. The value range is from `1` to `100`. The quality is effective only when using a lossy compression algorithm for the specific class.

MrcCompQuality

Property (get, set): double MrcCompQuality
Default: 10

Compression quality for MRC foreground and background images. The binary mask cannot be compressed. The value range is from 1 to 100. The quality is effective only when using a lossy compression algorithm (see table) for the specific class.

BinaryDownsampling

Property (get, set): double BinaryDownsampling
Default: 1.0

Down sampling factor for binary images. A value of 2 e.g. means that the resulting image dimension and resolution are divided by two.

GrayscaleDownsampling

Property (get, set): double GrayscaleDownsampling
Default: 1.0

Down sampling factor for grayscale images. A value of 2 e.g. means that the resulting image dimension and resolution are divided by two.

LabDownsampling

Property (get, set): double LabDownsampling
Default: 1.0

Down sampling factor for lab images. A value of 2 e.g. means that the resulting image dimension and resolution are divided by two.

Rgb_YcbcrDownsampling

Property (get, set): double Rgb_YcbcrDownsampling
Default: 1.0

Down sampling factor for RGB and YCbCr images. A value of 2 e.g. means that the resulting image dimension and resolution are divided by two.

CmykDownsampling

Property (get, set): double CmykDownsampling
Default: 1.0

Down sampling factor for CMYK images. A value of 2 e.g. means that the resulting image dimension and resolution are divided by two.

MrcDownsampling

Property (get, set): `double MrcDownsampling`
Default: `1.0`

Down sampling factor for MRC background and foreground images. The binary mask is not subjected to downsampling. A value of 2 e.g. means that the resulting image dimension and resolution are divided by two.

Recompress

Property (get, set): `BoolEnum Recompress`
Default: `1.0`

Recompress streams that have already been compressed with a lossy algorithm (JPEG, JPEG2000) in order to reduce the size by specifying either a different algorithm or a different quality.

PerformMrc

Property (get, set): `BoolEnum PerformMrc`
Default: `False`

Separate the image into background, foreground and mask layer.

BinarizationThreshold

Property (get, set): `double BinarizationThreshold`

The threshold value determines which pixel is associated to the foreground layer respectively to the background layer. (Black is `0`, white is `255`.) The threshold value should be set near to the average gray level of the image to achieve the best compression quality and ratio. The default threshold value is computed automatically from the image sample data.

UpgradeJpeg

Property (get, set): `BoolEnum UpgradeJpeg`
Default: `False`

Re-writes TIFF pages containing JPEG streams according to the TIFF V6 specification in the newer format conforming to the Technical Note #2 of the independent JPEG group. The re-formatting does not involve any re-compression and thus is lossless.

6.10.3 RecognizerParameters Struct

OcrEngineName

Property (get, set): `String OcrEngineName`
Default: `Nothing`

One of the supported OCR engines can be selected. Currently the values `"abbyy10"`, `"abbyy11"` and `"service"` are supported. In order to function correctly the tool requires the selected engine add-in to be installed.

In addition the value `"internal"` is supported. In this case no add-in is needed. For more information see Section 4.2.

OcrParameters

Property (get, set): `String OcrParameters`
Default: `Nothing`

Is an OCR engine specific string parameter. For more information on this, refer to the documentation of the engine add-in.

OcrLanguages

Property (get, set): `String OcrLanguages`
Default: `Nothing`

Receive a language identifier. The language is a hint for the OCR engine to recognize the text in the given language.

BinarizationPriorOcr

Property (get, set): `BoolEnum BinarizationPriorOcr`
Default: `False`

Convert color and gray scale image data to bi-tonal samples before sending the data to the OCR engine.

BinarizationThreshold

Property (get, set): `double BinarizationThreshold`
Default: `128`

Define the black/white threshold.

6.10.4 ImporterParameters Struct

Resolution

Property (get, set): `double Resolution`
Default: `96.0`

Recompress streams that have already been compressed with a lossy algorithm (JPEG, JPEG2000, JBIG2) in order to reduce the size by specifying either a different algorithm or a different quality.

UpgradeJpeg

Property (get, set): `BoolEnum UpgradeJpeg`
Default: `False`

Re-writes TIFF pages containing JPEG streams according to the TIFF V6 specification in the newer format conforming to the Technical Note #2 of the independent JPEG group. The re-formatting does not involve any re-compression and thus is lossless.

6.10.5 MergerParameters Struct

RemoveBlankPages

Property (get, set): `BoolEnum RemoveBlankPages`
Default: `False`

Pages that have been recognized by the `Recognizer` to be blank are removed from the TIFF file.

6.10.6 SplitterParameters Struct

PageNumberStart

Property (get, set): `int PageNumberStart`
Default: `FirstPage`

Start number of page selection range. If this propety is not set, by default the first page will be used.

PageNumberEnd

Property (get, set): `int PageNumberEnd`
Default: `LastPage`

End number of page selection range. If this property is not set, by default the last page will be used.

6.11 Enumerations

6.11.1 Conformance Enumeration

Unknown The conformance is unkown or automatically determined.

Pdf10 PDF Version 1.0

Pdf11 PDF Version 1.1

Pdf12 PDF Version 1.2

Pdf13 PDF Version 1.3

Pdf14 PDF Version 1.4 (corresponds to Acrobat 5)

Pdf15 PDF Version 1.5

Pdf16 PDF Version 1.6 (corresponds to Acrobat 7)

Pdf17 PDF Version 1.7

Pdf20 PDF Version 2.0

PdfA1B PDF/A-1b, ISO 19005-1, Level B conformance

PdfA1A PDF/A-1a, ISO 19005-1, Level A conformance

PdfA2B PDF/A-2b, ISO 19005-2, Level B conformance

PdfA2U PDF/A-2u, ISO 19005-2, Level U conformance

PdfA2A PDF/A-2a, ISO 19005-2, Level A conformance

PdfA3B PDF/A-3b, ISO 19005-3, Level B conformance

PdfA3U PDF/A-3u, ISO 19005-3, Level U conformance

PdfA3A PDF/A-3a, ISO 19005-3, Level A conformance

6.11.2 BoolEnum Enumeration

Compression_Undef Not defined.

True True.

False False.

6.11.3 Compression Enumeration

CompressionUndef Compression not defined.

CompressionUncompressed No compression.

CompressionCcittGroup31d CCITT Fax Group 3 1d.

CompressionCcittGroup3 CCITT Fax Group 3.

CompressionCcittGroup4 CCITT Fax Group 4.

CompressionLzw Lempel-Ziv-Welch.

CompressionJpeg Joint Photographic Expert Group. Lossy compression.

CompressionJpegTn2 Joint Photographic Expert Group Tech. Note #2. Lossy compression.

CompressionAdobeDeflate Adobe Defalte.

CompressionPackBits Pack Bits.

CompressionDeflate Deflate.

CompressionJpeg2000 Wavelet transformation. Lossy compression.

CompressionJbig2 Joint Bi-level Image Experts Group. Compression not lossy.

6.11.4 ErrorCode Enumeration

See Programming interfaces for more information about how these codes are mapped to exceptions in the .NET and Java interface.

Logic errors

These codes denote errors in the application program logic and should never happen at runtime.

UnsupportedOperation The requested method or property is not supported.

IllegalState The object is in a state where the requested object or property cannot be called.

IllegalArgument The method was called using an illegal argument.

Environmental errors

Generic The error is not further specified.

Fatal A fatal error occurred.

License Licensing error.

NotFound The requested item or resource could not be found.

IO Error while reading or writing from a stream.

UnknownFormat The format is unknown.

Corrupt The data is corrupt.

Password The resource or document is protected by a password.

Conformance A conformance mismatch happened.

UnsupportedFeature The file contains an unsupported feature.

Infrastructure An infrastructure error occurred.

Processing The file cannot be processed.

Exists The item already exists.

7 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.

7.1 Changes in versions 6.19–6.27

- **Update** license agreement to version 2.9

7.2 Changes in versions 6.13–6.18

No functional changes.

7.3 Changes in versions 6.1–6.12

- [.NET] **Improved** `ErrorCodeException` which is now serializable (except for the .NET Standard 1.0 target).
- [Java] **Changed** minimal supported Java language version to 7 [previously 6].
- [.NET] **New** availability of this product as NuGet package for Windows.
- [.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.
- [Java] **Changed** inheritance. All interfaces except `StringList` now inherit from `AutoCloseable` and therefore can be used in a try-resource clause.
- [Java] **New** static method `fromValue` for all non-flags enums.

Interface `StringList`

- [Java] **Deprecated** method `close`.
- [.NET] **Deprecated** method `Dispose`.

7.4 Changes in version 5

- **New** additional supported operating system: Windows Server 2019.
- **Changed** behavior when reading a TIFF. The value `Relative` from tag `ResolutionUnit` is now interpreted as `Inch`.
- [.NET] **Changed** `TiffToolboxNET.dll` library. Cross-product functionality is outsourced into common library `PdfCommonNET.dll`.

Interface `Compressor`

- **Improved** MRC background and foreground layer coloring.

7.5 Changes in version 4.12

- **New** HTTP proxy setting in the GUI license manager.
- **New** Interface `Scanner` to scan images from a TWAIN or WIA scanner.

Interface `Recognizer`

- **New** key `BlankPageMargin` for the formation of the `OcrParameters` property in struct `RecognizerParameters`. The corresponding value denotes the relative margin. The margin is excluded from the analysis if a page is blank.

Interface `Scanner`

- **New** method `SetSource` to set TWAIN or WIA scanner to scan with.
- **New** method `ShowUI` to show user interface that allows to configure scan settings.
- **New** method `Scan` to scan into output stream.
- **New** property `Capabilities` to get and set capability stream.
- **New** property `Sources` to get list (see `StringList`) of available TWAIN/WIA sources.

7.6 Changes in version 4.11

- **New** support for reading and writing PDF 2.0 documents.

7.7 Changes in version 4.10

- [C] **Clarified** Error handling of `TPdfStreamDescriptor` functions.

7.8 Changes in version 4.9

- **Improved** metadata generation for standard PDF properties.
- [C] **Changed** return value `pfGetLength` of `TPDFStreamDescriptor` to `pos_t`[5].

Interface `Converter`

- [.NET, C, Java] **New** method `SetDefaultColorSpace`: This color space is used in the output PDF for pages that have no ICC profile available in the corresponding page in the input TIFF. A default color space profile can be set for both RGB and CMYK.
- [Java] **New**: MemoryStream implementation for Stream interface.

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

7.9 Changes in version 4.8

All interfaces

- [.NET, C, Java] **New** property `ProductVersion` to identify the product version.
- [.NET] **New** static property setter `LicenseKey`.

Interface `Compressor`

- [.NET, C, Java] **New** method `SetOutputIntent` to set the output intent.

8 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.

Licensing and copyright The 3-Heights® TIFF Toolbox API is copyrighted. This user manual is also copyright protected; It may be copied and distributed provided that it remains unchanged including the copyright notice.

Contact

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