User Manual

3-Heights™ PDF Viewer API R2

Version 4.12.26.6

PDF-TOOLS.COM
Premium PDF Technology

Contents

1 Introduction

1.1 Description

The 3-Heights™PDF Viewer API R2 is a compact, high-performance, high-quality PDF viewer. It offers a multitude of navigational and display options for displaying documents. This viewer is mainly characterized by its increased performance and its uniform and simple interface.

1.2 Features

- Navigate manually (user action) or programmatically through a document
- Select between different fit modes: actual size, fit to width, fit to height
- Rotate and display the page
- Show thumbnails and use them for navigation
- Show outlines (Bookmarks)
- Create, edit and delete annotations (Java)
- Enter password to decrypt PDF documents
- Read document from file or memory
- Supports the Forms Data Format (FDF) file format

1.3 Supported Input Formats

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

1.4 Compliance

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 19005-2 (PDF/A-3)

1.5 Interfaces

There is a Java Swing, a .NET WPF and .NET Windows Forms interface available. The .NET interfaces use .NET framework version 4.0. However the use of .Net framework version 4.5.2 or newer is recommended, as the support lifecycle for older versions has ended.

1.6 Operating Systems

The 3-Heights™ PDF Viewer API R2 is available for the following operating systems:

- Windows 7, 8, 8.1, 10 – 32 and 64 bit

- Windows Server 2008, 2008 R2, 2012, 2012 R2, 2016 – 32 and 64 bit

1.7 How to Best Read this Manual

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

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

2 Installation and Deployment

2.1 Windows

The 3-Heights™ PDF Viewer API R2 comes as a ZIP archive.

The installation of the software requires the following steps.

1. You need administrator rights to install this software.
2. Log in to your download account at http://www.pdf-tools.com. Select the product "PDF Viewer API R2". If you have no active downloads available or cannot log in, please contact pdfsales@pdf-tools.com for assistance.
   You will find different versions of the product available. We suggest to download the version, which is selected by default. If another is required, it can be selected using the combo box.
   The product comes as a ZIP archive containing all files.
   There are 32 and 64-bit versions of the product available. While the 32-bit version runs on both, 32 and 64-bit platforms, the 64-bit version runs on 64-bit platforms only. The ZIP file contains both the 32-bit and the 64-bit version of the product.
3. Unzip the archive to a local folder, e.g. `C:\Program Files\PDF Tools AG\`.
   This creates the following subdirectories:

Subdirectory	Description
`bin`	Contains the runtime executable binaries.
`doc`	Contains documentation.
`jar`	Contains Java archive files for Java components.
`samples`	Contains sample programs in various programming languages

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

2.2 Interfaces

The 3-Heights™ PDF Viewer API R2 provides two different interfaces. The installation and deployment of the software depend on the interface you are using. The table below shows the supported interfaces and examples with which programming languages they can be used.

Interface	Programming Languages
.NET	The MS software platform .NET can be used with any .NET capable programming language such as: ■ C# ■ VB .NET ■ J# ■ others This interface is available in the Windows version only.
Java	The Java interface.

2.2.1 Development

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

Req. This file is required for this interface.

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

Doc. This file is for documentation only.

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

Files for Development

Name	.NET	Java
bin\‹platform›\PdfViewerAPI.dll	Req.	Req.
bin*NET.dll	Req.	
bin*NET.xml	Doc.	
doc*.pdf	Doc.	Doc.
doc\javadoc*.*		Doc.
jar\PdfViewerAWT.jar		Req.
samples*.*	Doc.	Doc.

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

File Description

Name	Description
bin\‹platform›\PdfViewerAPI.dll	This is the DLL that contains the main functionality (required).

File Description

`bin\*NET.dll`	The .NET assemblies are required when using the .NET interface. The files `bin\*NET.xml` contain the corresponding XML documentation for MS Visual Studio.
`doc\*.*`	Various documentations.
`jar\PdfViewerAWT.jar`	The Java API archive.
`samples\*.*`	Contains sample programs in different programming languages.

2.2.2 Deployment

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

Files for Deployment

Name	.NET	Java
`bin\‹platform›\PdfViewerAPI.dll`	Req.	Req.
`bin\*NET.dll`	Req.	
`jar\PdfViewerAWT.jar`		Req.

The deployment of an application works as described below:

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

2.3 Interface Specific Installation Steps

2.3.1 Java Interface

The 3-Heights™ PDF Viewer API R2 requires Java version 6 or higher.

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

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

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

For execution Additionally the library `PdfViewerAPI.dll` needs be in one of the system's library directories[2] or added to the Java system property `java.library.path`. This can be achieved by either adding it dynamically at program startup before using the API, or by specifying it using the switch `-Djava.library.path` when starting the Java VM. Choose the correct subdirectory `x64` or `Win32` depending on the platform of the Java VM[3].

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

2.3.2 .NET Interface

The 3-Heights™ PDF Viewer API R2 does not provide a pure .NET solution. Instead, it consists of .NET assemblies, which are added to the project and a native DLL, which is called by the .NET assemblies. This has to be accounted for when installing and deploying the tool.

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

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

The native DLL `PdfViewerAPI.dll` is called by the .NET assembly `PdfViewerNET.dll`.

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

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

2.4 Color Profiles

For calibrated color spaces (such color spaces with an associated ICC color profile) the color conversion is well defined. For the conversion of uncalibrated device color spaces (DeviceGray, DeviceRGB, DeviceCMYK) however, the 3-Heights™ PDF Viewer API R2 requires apropriate color profiles. Therefore it is important, that the profiles are available and that they describe the colors of the device your input documents are intended for.

If no color profiles are available, default profiles for both RGB and CMYK are generated on the fly by the 3-Heights™ PDF Viewer API R2.

2.4.1 Default Color Profiles

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

Windows

1. `%SystemRoot%\System32\spool\drivers\color`
2. directory `Icc`, which must be a direct sub-directory of where the `PdfViewerAPI.dll` resides.

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

2.4.2 Get Other Color Profiles

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

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

2.5 Fonts

2.5.1 Font Cache

A cache of all fonts in all Font Directories is created. If fonts are added or removed from the font directories, the cache is updated automatically.

In order to achieve optimal performance, make sure that the cache directory is writable for the 3-Heights™ PDF Viewer API R2. Otherwise the font cache cannot be updated and the font directories have to be scanned on each program startup.

The font cache is created in the subdirectory `<CacheDirectory>/Installed Fonts` of the Cache Directory.

2.6 Note about the Evaluation License

With the evaluation license the 3-Heights™ PDF Viewer API R2 automatically adds a watermark to the displayed pages.

2.7 Special Directories

2.7.1 Directory for temporary files

This directory for temporary files is used for data specific to one instance of a program. The data is not shared between different invocations and deleted after termination of the program.

The directory is determined as follows. The product checks for the existence of environment variables in the following order and uses the first path found:

Windows

1. The path specified by the `%TMP%` environment variable.
2. The path specified by the `%TEMP%` environment variable.
3. The path specified by the `%USERPROFILE%` environment variable.
4. The Windows directory.

2.7.2 Cache Directory

The cache directory is used for data that is persisted and shared between different invocations of a program. The actual caches are created in subdirectories. The content of this directory can safely be deleted to clean all caches.

This directory should be writable by the application, otherwise caches cannot be created or updated and performance will degrade significantly.

Windows

- If the user has a profile:
  `%LOCAL_APPDATA%\PDF Tools AG\Caches`
- If the user has no profile:
  `<TempDirectory>\PDF Tools AG\Caches`

where `<TempDirectory>` refers to the Directory for temporary files.

2.7.3 Font Directories

The location of the font directories depends on the operating system. Font directories are traversed recursively in the order as specified below.

If two fonts with the same name are found, the latter one takes precedence, i.e. user fonts will always take precedence over system fonts.

Windows

1. `%SystemRoot%\Fonts`
2. directory `Fonts`, which must be a direct sub-directory of where `PdfViewerAPI.dll` resides.

3 License Management

The 3-Heights™ PDF Viewer API R2 requires a valid license in order to run correctly. If no license key is set or the license is not valid, then most of the interface elements documented in Interface Reference will fail with an error code and error message indicating the reason.

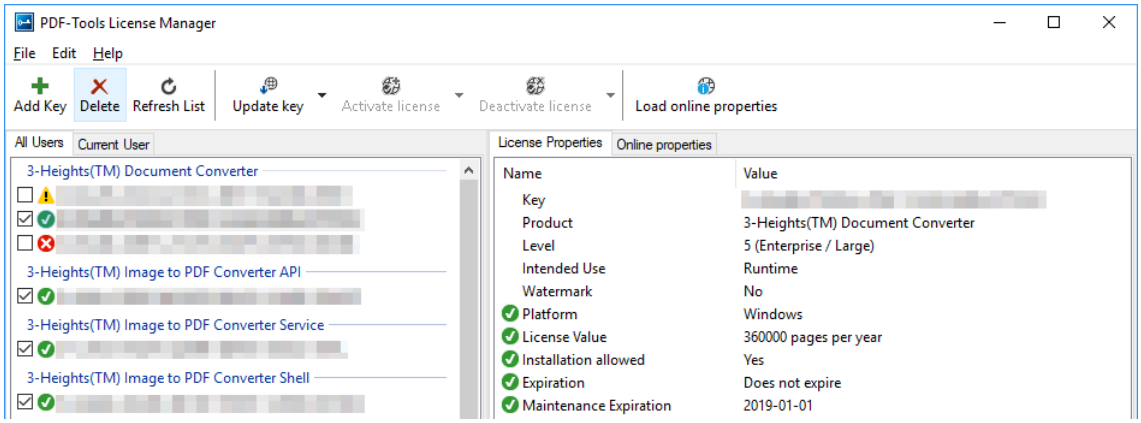
3.1 License Installation and Management

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

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

3.1.1 Graphical License Manager Tool

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



List all installed license keys

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

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

Add and delete license keys

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

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

Display the properties of a license

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

3.1.2 Command Line License Manager Tool

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

> **Note:** The command line tool licmgr is not included in Windows platform kits, as the GUI tool is the recommended tool for managing Licenses. A Windows licmgr shelltool is available on request.

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

```
licmgr
```

List all installed license keys

```
licmgr list
```

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

Example:

```
>licmgr list
Local machine:
  Product Name:
      1-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
      1-YYYYY-YYYYY-YYYYY-YYYYY-YYYYY-YYYYY-YYYYY
    * 1-ZZZZZ-ZZZZZ-ZZZZZ-ZZZZZ-ZZZZZ-ZZZZZ-ZZZZZ
Current user:
```

Add and delete license keys

Install new license key:

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

Delete old license key:

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

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

g For all users

u Current user

Display the properties of a license

```
licmgr info 1-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

Properties that invalidate the license are marked with an X, properties that require attention are marked with an !. In that case an additional line with a comment is displayed.

Example:

```
>licmgr info 1-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
 - Key:          1-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
 - Product:      Product Name
 - Features:     Feature1,Feature2
 - Intended use: Development
 - Watermark:    No
 - Platform:     Windows
 - Installation: Yes
 ! Activation:   2018-05-07
                  (The license has not yet been activated.)
 - Expiration:   Does not expire
 - Maintenance:  2019-04-27
```

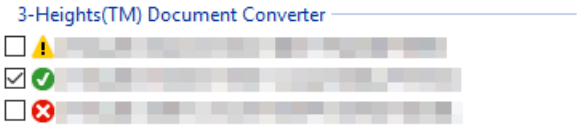
3.2 License Selection and Precedence

3.2.1 Selection

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

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

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



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

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

3.2.2 Precedence

License keys are considered in the following order:

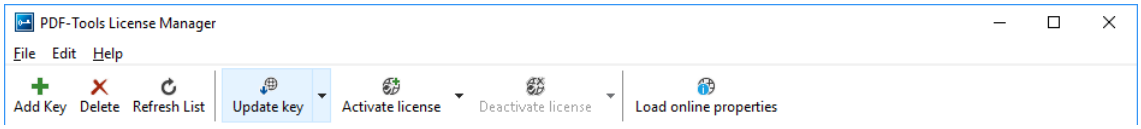
1. License key passed at runtime.
2. License selected for the current user
3. License selected for the current user (legacy key format)
4. License selected for all users
5. License selected for all users (legacy key format)

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

3.3 Key Update

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

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



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

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

3.4 License activation

New licenses keys have to be activated (except for OEM licenses).

> **Note:** Licenses that need activation have to be installed in the license manager and must not be passed to the component at runtime.

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

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

3.4.1 Activation

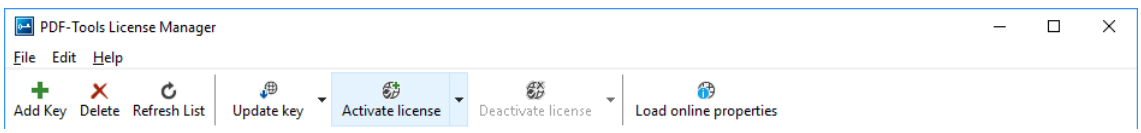
The License can be activated directly within the license manager. Every activation increases the activation count of the license by 1.

It is recommended to add a comment to the activation request which helps keeping track of all activations for a specific license key. In case of problems it also helps us providing support.

The comment is stored in the activation database as long as the license key remains activated. Upon deactivation it is deleted from the database immediately.

All activations and the corresponding comments can be examined using the **Load online properties** function of the license manager. The information is accessible to anyone with access to the license key.

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



It is recommended to add a comment to the activation request by using the subsequent dialog box.

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

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

Note that the key has to be installed first.

It is recommended to add a comment to the activation request by using the `-c` or `-cd` option:

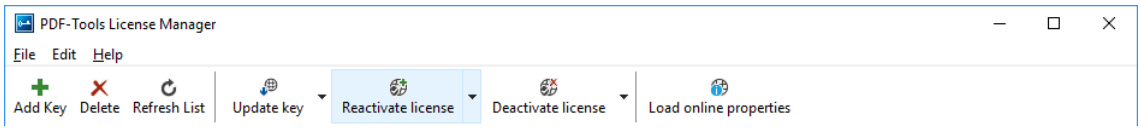
```
licmgr activate -cd 1-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
licmgr activate -c "custom comment" 1-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

3.4.2 Reactivation

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

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

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



With the Command Line Interface the subcommand `activate` is used again:

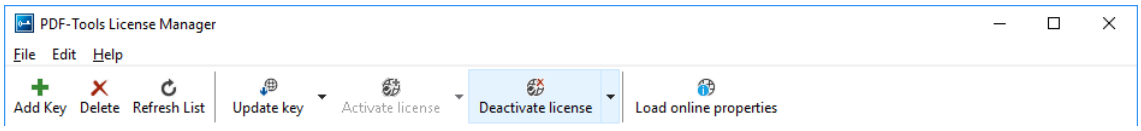
```
licmgr activate 1-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

3.4.3 Deactivation

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

The process for deactivation is similar to the activation process.

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



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

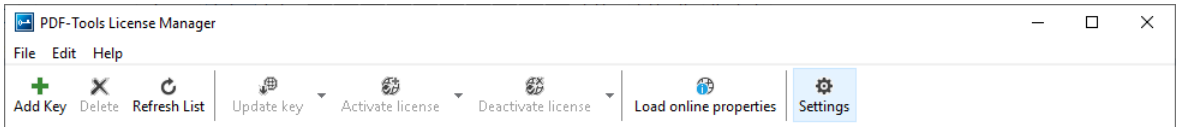
```
licmgr deactivate 1-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

3.5 Proxy Setting

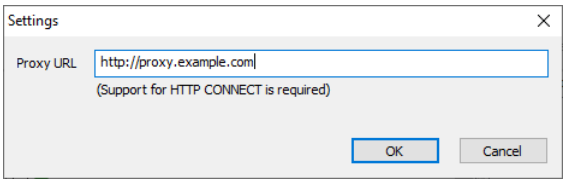
A proxy URL can be configured for computers that cannot access the internet without a web proxy.

Note: The proxy must allow connections via HTTP CONNECT to the server www.pdf-tools.com:443.

In the Graphical User Interface press the button "Settings" in the toolbar:



and enter the proxy URL in the respective field:



3.6 Offline Usage

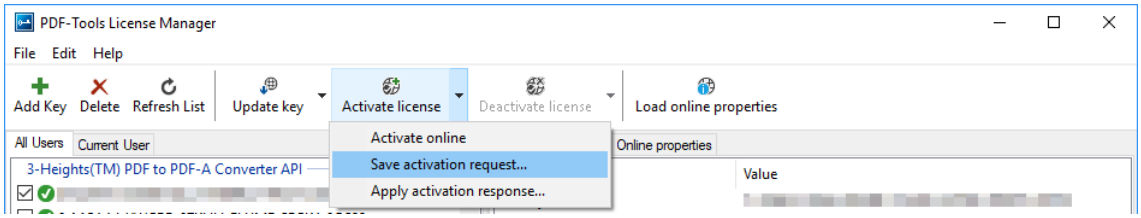
The following actions in the license manager need access to the internet:

- License Activation
- License Reactivation
- License Deactivation
- Key Update

On systems without internet access, a three step process can be used instead, using a form on the PDF Tools website.

3.6.1 First Step: Create a Request File

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



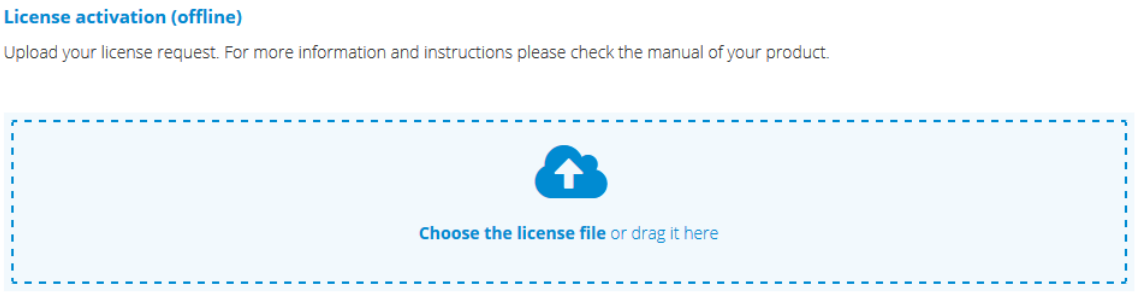
With the Command Line Interface use the `-fs` option to specify the destination path of the request file:

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

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

3.6.2 Second Step: Use Form on Website

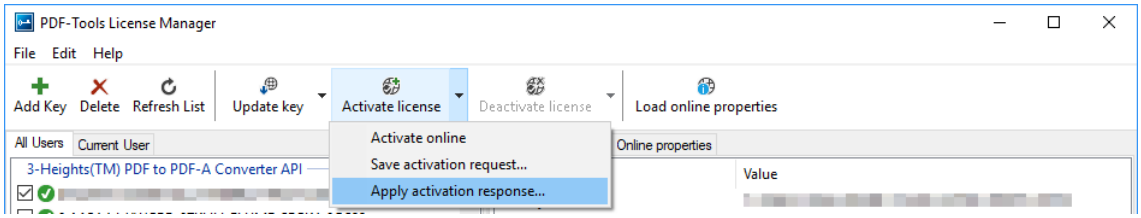
Open the following website in a web browser: http://www.pdf-tools.com/pdf20/en/mypdftools/licenses-kits/license-activation/ Upload the request by dragging it onto the marked area:



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

3.6.3 Third Step: Apply the Response File

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



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

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

3.7 License Key Versions

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

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

3.8 License Key Storage

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

3.8.1 Windows

The license keys are stored in the registry:

- "`HKLM\Software\PDF Tools AG`" (for all users)

- "HKCU\Software\PDF Tools AG" (for the current user)

3.9 Troubleshooting

3.9.1 License key cannot be installed

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

Possible causes:

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

Solution

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

3.9.2 License is not visible in license manager

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

Possible causes:

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

Solution

Use a current version of the license manager application.

3.9.3 License is not found at runtime

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

Possible causes:

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

Solution

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

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

3.9.4 Eval watermark is displayed where it should not

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

Possible causes:

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

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

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

Solution

Disable or remove all evaluation license in all scopes, check that no evaluation key is passed at runtime and restart the software.

3.9.5 Activation is not recognized

The license is installed and activated in the license manager, but the software does not recognize it as activated.

The error message is: `"The license has not been activated."`

Possible causes:

- There is an unregistered license key selected for the **current user**, that takes precedence over the key for **all users**. This leads to an error even if the same license is registered for all users.

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

- A license key that is passed at runtime takes precedence over those selected in the license manager. This leads to an error even if the same license is registered in the license manager.

> **Note:** Licenses that need activation have to be installed in the license manager and must not be passed to the component at runtime.

- The software was not restarted after activating the license.

Solution

Disable, remove or activate all unregistered licenses in all scopes, check that no key is passed at runtime and restart the software.

3.9.6 Activation is invalidated too often

The license activation is invalidated regularly, for no obvious reason.

Possible causes:

- The MAC address used for computing the machine fingerprint is not static. This may happen e.g. for virtual network adapters with dynamic MAC address (VPN, Juniper, ...).

Solution

Update to a newer version (>= 4.12) of the PDF Tools product, deactivate the license key using the new license manager and activate it again. After that, an imporved fingerprinting algorithm is used.

Deactivation and activation have to be **executed separately**, a reactivation of the license in one step does not change the fingerprinting algorithm and thus does not solve the problem.

> **Note:** After this procedure, older products might not recognize the activation as valid anymore. Reactivating the license using an old license manager will revert the activation to the old fingerprinting algorithm.

As an alternative, remove any virtual network adapter with a dynamic MAC address.

3.9.7 Connection to the licensing service fails

The license activation/deactivation/update fails because the license manager cannot reach the licensing server.

The error message depends on the platform and the exact error condition.

Possible causes:

- The computer is not connected to the internet.
- The connection is blocked by a corporate firewall.

Solution

Make sure that the computer is connected to the internet and that the host `www.pdf-tools.com` is reachable on port 443 (HTTPS).

If this is not possible, try Offline Usage instead.

3.9.8 Offline usage fails due to a request/response mismatch

The offline license activation/deactivation/update fails because the response file does not match the request file.

The error message is: `"Mismatch between request and response."`

Possible causes:

- The response file is applied to a different machine than the request file was created.
- The response file as applied to a different user than the request file was created.
- The response file was applied to a specific user while the request was created for all users, or vice versa.
- The response file is applied to the wrong license key.
- Another request file has been created between creating the request file and applying the response file.
- The license key was updated between creating the request file and applying the response file.
- The license key was removed and re-added between creating the request file and applying the response file.

Solution

Delete any old request and response files to make sure they are not used by accident.

Retry the entire process as outlined in chapter 3.6 and refrain from making any other license-related actions between creating the request file and applying the response file.

Make sure that the response file is applied to exactly the same license key in exactly the same location (machine, all users or specific user) where the request file was created.

4 Programming Interfaces

4.1 .NET

The two .NET interfaces for WPF and Windows Forms do not require any further installation. Simply reference the DLLs in the project of your application. Both, `bin\‹platform›\PdfViewerAPI.dll` and `bin\PdfViewerCSharpAPI.dll` are required for .NET interfaces. In addition either `bin\PdfViewerWPF.dll` or `bin\PdfViewerWinForms.dll` is required, depending on which interface is being used.

Note, that referencing the `.xml` documentation files contained in the `bin\` folder can also be useful when using an Integrated Development Environment (IDE). They contain additional documentation information about the interface that will be displayed to the user by the IDE.

Do ensure to use .NET Framework version 4.0 or newer for the application that uses the dll's.

After installing and registering the 3-Heights™ PDF Viewer API R2, you find .NET sample solutions in the path `samples\CS.NET\`. There are samples provided for .NET WPF as well as .NET Windows Forms. For each interface there is both a minimal example, showing how to interface with the API to achieve basic functionality (i.e. opening a file and showing it) as well as an advanced example, with a fully integrated UI that can be used like a normal desktop viewer. For each sample, a solution file is provided.

For starting out with the integration of the PDF Viewer API R2, it is recommended to use the minimal sample as a reference to achieve basic functionality. For integrating advanced features later, one may consult the advanced sample, which shows at least one way of integrating each feature.

To run the provided samples, ensure that you have registered your license using the License Manager (see Chapter 3 License Management) and that the platform of the kit you downloaded matches the platform you have configured the IDE to use (64 bit vs. 32 bit).

When implementing the API in your own application, consider that the API uses .NET framework 4.0. This means that the integrating application has to use .NET framework 4.0 or newer.

4.2 Java

The available Java Interface uses Swing and does not require any further installation. Simply reference the Java archive file `jar\PdfViewerAWT.jar` as well as the native library `bin\‹platform›\PdfViewerAPI.dll` for building your Java project.

Note that in most Integrated Development Environments (IDEs) it is possible to link the provided javadoc in `doc\javadoc\*` to the Java archive file `jar\PdfViewerAWT.jar`. Alternatively one can also view the javadoc directly in a browser by opening `doc\javadoc\index.html`.

Be aware that both Java as well as the PDF Viewer API R2 are platform specific (32 or 64 bit). This means that the 64 bit version of the PDF Viewer API R2 will only work with 64 bit Java installations.

After installing and registering the PDF Viewer API R2, you find Java samples in the path `samples\java\`. The provided samples are written in Swing and AWT. There is a minimal sample, showing how to interface with the API to achieve basic functionality (i.e. opening a file and showing it). The second sample uses all the advanced features that the API offers and has a fully integrated UI that can be used like a normal desktop viewer.

For each sample a `run.bat` script is provided, which compiles and runs the sample using the Java installation as configured by the PATH variable.

For starting out with the integration of the PDF Viewer API R2, it is recommended to use the minimal sample as a reference to achieve basic functionality. For integrating advanced features later, one may consult the advanced sample, which shows at least one way of integrating each feature.

5 User's Guide

Most samples in this guide are written in C#. The call sequence, however, for Java is the same.

5.1 Open a PDF Document

Documents can be opened either from file using the method Open or from memory using the method OpenMem.

Example: Open PDF from File System.

```
private void Open(){
  viewer.onOpenCompleted += OnOpenCompletedEventHandler;
  viewer.Open(Directory.GetCurrentDirectory() + "\input.pdf", "");
}
private void OnOpenCompletedEventHandler(PdfViewerException ex)
{
  if(ex != null)
    MessageBox.Show("Failed to open file: " + ex.getMessage());
  viewer.onOpenCompleted -= OnOpenCompletedEventHandler;
}
```

Viewer is an object of type PdfViewerWPF or PdfViewerWinForms. The sample checks whether opening the input file was successful or not. If not, it shows a message box.

Note that it is required to register an eventHandler, as the opening operation is asynchronous and the **viewer.Open** command immediatly returns. In general it is recommended that you attach the eventHandler once at the beginning and don't detach when it gets triggered. However, if multiple sources use the open command and require their own event handlers, then attaching and detaching as shown in the example might be useful.

OpenMem is usually used when a PDF document is already available in memory, e.g. is read from a data base or is passed in-memory from another application.

The following example shows how a PDF document can be opened form memory by reading it from file and writing it in a byte array and then reading that byte array using OpenMem.

Example: Open PDF from Memory.

```
private void OpenMem_Click()
{
  viewer.onOpenCompleted += OnOpenCompletedEventHandler;
  byte[] fileMem = File.ReadAllBytes
    (Directory.GetCurrentDirectory() + "\input.pdf");
  viewerForm.viewerAPI.OpenMem(fileMem, "");
}
private void OnOpenCompletedEventHandler(PdfViewerException ex)
{
  if(ex != null)
    MessageBox.Show("Failed to open file: " + ex.getMessage());
  viewer.onOpenCompleted -= OnOpenCompletedEventHandler;
}
```

Using OpenMem is especially useful if the file is already in memory. This can either happen when the application just created the file or if the same file is used multiple times.

5.2 Navigation

There are various ways how the user can navigate in a document with the PDF Viewer API R2. The following features can be used to navigate:

- Use the property PageNo to set page number to be displayed. The page range goes from 1 to PageCount.
- Use the mouse in cursor mode eMouseMoveMode (default) to scroll the page by pressing the left mouse button and moving the mouse up and down. Change the cursor mode using the property MouseMode.
- Use horizontal and vertical scroll bars.
- Use the mouse wheel to scroll vertically or hold the control key to zoom in and out with the mouse wheel.
- Use the property Destination to move to a specific location and zoom level within the document.
- Use the outlines (bookmarks) or thumbnails from the navigation panel at left hand side of the control to access any page or section.
  - The outlines and thumbnails panels can be enabled or disabled using the properties ShowOutlines and ShowThumbnails.
- The Search function can be used to move the viewport to the found match.

5.3 Suspend Layouting

The methods SuspendLayout allows the layouting to be suspended. This can be useful if the implementing application wants to change multiple visual settings, but wishes to avoid having multiple visual updates. The layouting can then later be resumed using ResumeLayout.

The following example shows an advanced application of this principle. It suspends layouting before opening a file and then waits for the file having completed opening, using the OpenCompleted event. It then applies some custom navigation commands and only then resumes layouting. For the user this results in one single visual update, whereas without suspending, he might catch a few glimpses of the intermediate states in between the visual updates.

Example: Suspend layouting to prevent multiple visual updates.

```
private void Open(String fileName)
{
  viewer.SuspendLayout();
  viewer.OpenCompleted += OnOpenCompletedHandler;
  viewer.Open(fileName, "");
}

private void OnOpenCompletedHandler(PdfViewerException exception)
{
  viewer.OpenCompleted -= OnOpenCompletedHandler;
  if(exception == null)
  {
    viewer.PageLayoutMode = TPageLayoutMode.SinglePage;
    viewer.PageNo = 1;
    viewer.Rotate = 90;
  }
  viewer.ResumeLayout();
}
```

5.4 Using viewer in background

The PDF Viewer API R2 does not require to have a visual parent to be used. This means that one can instantiate the control in the background, open a file and only display it to the user once it is loaded. This is illustrated in the following example, assuming that the **this** object is a WPF control.

Example: Add viewer only to visual control after it has successfully opened the file.

```
private void Open(String fileName)
{
  viewer.OpenCompleted += OnOpenCompletedHandler;
  viewer.Open(fileName, "");
}

private void OnOpenCompletedHandler(PdfViewerException exception)
{
  viewer.OpenCompleted -= OnOpenCompletedHandler;
  if(exception == null)
  {
    this.Children.Add(viewer);
  }
  viewer.ResumeLayout();
}
```

5.5 Listening to Property changes

The API offers a multitude of properties that describe the current state of the viewer (see section Properties). However the application might need to not only know the current state, but also be informed when that state changes. For this purpose, the PDF Viewer API R2 implements the INotifyPropertyChanged Interface and the corresponding `PropertyChanged` event. The `PropertyChanged` event passes the string of the property that just changed along. This is illustrated in the following sample, which will listen for changes of the `FitMode` property and highlight toolstripbuttons as checked accordingly.

Example: Listen to changes of active fitmode.

```
viewer.PropertyChanged += OnFitModeChanged;

private void OnFitModeChanged(object sender, PropertyChangedEventArgs e)
{
  if (!e.PropertyName.Equals("FitMode"))
    return;
  fitPage.Checked = (viewerAPI.FitMode ==
    PdfTools.PdfViewerCSharpAPI.Model.FitMode.FitPage);
  fitWidth.Checked = (viewerAPI.FitMode ==
    PdfTools.PdfViewerCSharpAPI.Model.FitMode.FitWidth);
  fitActualSize.Checked = (viewerAPI.FitMode ==
    PdfTools.PdfViewerCSharpAPI.Model.FitMode.FitTrueSize);
}
```

This same pattern can be used with almost all readable properties, such as `Zoom`, `PageNo` or `MouseMode`.

When using the WPF API, then most properties are also available as dependency properties, which can be bound to directly in the xaml markup.

5.6 Printing

Currently printing is only supported for the Java API.

The printing is achieved by using a class that implements the `Printable` interface. The `print` function of the printing class can make use of the method `drawSinglePage()`. This method returns a `BufferedImage` which can then be used to be drawn in the `Graphics` context provided by the `print()` parameters.

Example: Overview of the printer class

```
@Override
public int print(Graphics graphics, PageFormat pageFormat, int pageIndex)
{
  ...
  // viewer is of type PdfViewerComponent
  BufferedImage image = viewer.drawSinglePage(...);
  ...
  graphics.drawImage(image, ...);

}
```

The printing is synchronous this means large documents will block the viewer while the pages are being rendered. Also the memory consumption can be high during printing but should go back to normal levels after the print job has finished.

6 Interface Reference

In the following sections, we document the interface of the PdfViewerAPI, when using C#. Both a WPF and a Windows Forms Interface are available. While they are mostly identical, there are a few key differences, imposed by the used graphical library. For example, WPF allows the use of Dependency Properties, so all public properties in the WPF interface can be used as bindings and thus notify changes of the property value automatically. In Windows Forms, the viewer client has to implement the INotifyPropertyChanged Interface and must then manually handle the PropertyChanged Event to update these values.

The java interface is not described in detail, however the java interface is very similiar to the here described interfaces. Java language limitations have required some changes though:

- Each C# property corresponds to a pair of get and set methods in java. (Example: C# Property `Zoom` corresponds to java methods **getZoomFactor** and **setZoomFactor**)
- Each C# event corresponds to a java Interface, with a single method. Classes implementing this interface are called **EventListeners** and you can then register the listener on the API (Example: C# event `SearchCompleted` corresponds to java interface IOnSearchCompletedListener with method onSearchCompleted and can be registered at PdfViewerComponent.registerOnSearchCompleted)

6.1 Enumerations

The following enumerations are defined in
`PdfTools.PdfViewerCSharpAPI.Model.IPdfViewerController`

> **Java API Note:** In Java these definitions are found in
> `com.pdf_tools.pdfviewer.Model.IPdfViewerController`

6.1.1 TPageLayoutMode Enumeration

TPageLayoutMode Table

TPageLayoutMode	
`SinglePage`	Shows one single page, scrolling when reaching the top/bottom of the page will jump to the next/previous page.
`OneColumn`	Shows one column of pages, that can be scrolled through.
`TwoColumnLeft`	Shows a column of pairs of pages, that can be scrolled through. The first page is placed left.
`TwoColumnRight`	Shows a column of pairs of pages, that can be scrolled through. The first page is placed as a separate title page at the top.
`TwoPageLeft`	Shows a pair of pages, scrolling when reaching the top/bottom of the page will jump to the next/previous page pair. The first page is placed left.

TPageLayoutMode Table

TwoPageRight	Shows a pair of pages, scrolling when reaching the top/bottom of the page will jump to the next/previous page pair. The first page is placed as a separate title page at the top.

6.1.2 FitMode Enumeration

FitMode Table

FitMode	
FitPage	The window is zoomed to fit the whole page into the viewport.
FitWidth	The window is zoomed to fit the page's width into the viewport. If there are multiple columns of pages shown, the viewport will fit to that width.
FitTrueSize	The window is zoomed to reflect the true size of the page.

6.1.3 TDestination Enumeration

TDestination

TDestination	
eDestinationInvalid	Placeholder for invalid Destinations.
eDestinationFit	Fits the viewport to the destination page
eDestinationFitH	Fits the viewport to the horizontal dimension of the destination page
eDestinationFitV	Fits the viewport to the vertical dimension of the destination page
eDestinationFitR	Fits the viewport to the destination rectangle
eDestinationFitB	Fits the viewport to the bounding box of the destination page
eDestinationFitBH	Fits the viewport to the horizontal dimension of the destination page bounding box
eDestinationFitBV	Fits the viewport to the vertical dimension of the destination page bounding box
eDestinationFitXYZ	Fits the viewport to be at a specific location with a specific zoom level

6.1.4 TMouseMode Enumeration

TMouseMode

TMouseMode	
eMouseUndefMode	Placeholder for undefined mousmode.
eMouseMoveMode	Use the mouse to move the viewport on the document.
eMouseZoomMode	Use the mouse to draw a rectangle. The viewport will then zoom in to fit said rectangle.
eMouseTextExtractMode	Use the mouse to draw a rectangle. All contained text will be extracted and returned as the event TextExtracted.
eMouseHighlightMode	Use the mouse to draw a rectangle. All text fragments contained within said rectangle will be highlighted.

6.1.5 TViewerTab Enumeration

TViewerTab

TViewerTab	
eOutlineTab	Select the outline tab (bookmarks tab).
eThumbnailTab	Select the thumbnails tab.

7 PDF Viewer Application Program Interface

In this section there is a detailed interface description of the .NET WPF Control **PdfTools.PdfViewerWPF.PdfViewerWPF**. The interface of the .NET Windows Forms Control **PdfTools.PdfViewerWinForms.PdfViewerWinForms** is very similar, with a few key differences imposed by Windows Forms. For the java swing Viewer Component **com.pdf_tools.pdfviewer.SwingAPI.PdfViewerComponent** the interface is somewhat different. Key differences are indicated in the following sections.

7.1 Properties

Note that this is the documentation of the .NET interface. In java properties are implemented as simple set and get methods. E.g. the equivalent of property **Rotate** in java are the methods **setRotation** and **getRotation**. For exact method names, consult the provided JavaDoc.

For the WPF interface most of these properties are also available as dependency properties with the same name.

7.1.1 PageCount

Property (get): `int PageCount`

The number of pages in the opened document. 0 if no Document open.

7.1.2 Destination

Property (get, set): `PdfTools.PdfViewerCSharpAPI.Utilities.Destination Destination`

Get or set the currently displayed destination.

7.1.3 SelectedViewerTab

Property (get, set): `PdfTools.PdfViewerCSharpAPI.Model.TViewerTab SelectedViewerTab`

Get or set the currently selected tab on the sidepanel (Outlines or thumbnails). See TViewerTab.

7.1.4 Rotate

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

The “rotate” angle in multiples of 90 degrees. Each individual page is rotated by the given angle.

7.1.5 Border

Property (get, set): `double Border`
Default: `6.0`

The border displayed in between pages in user units.

7.1.6 Resolution

Property (get, set): `PdfTools.PdfViewerCSharpAPI.Model.Resolution Resolution`
Default: `(96,96)`

The display resolution in DPI.

7.1.7 FitMode

Property (get, set): `PdfTools.PdfViewerCSharpAPI.Model.FitMode FitMode`
Default: `FitDocument`

The FitMode used for displaying the document. See also `FitMode`.

7.1.8 PageLayoutMode

Property (get, set): `PdfTools.PdfViewerCSharpAPI.Model.TPageLayoutMode PageLayoutMode`
Default: `PageLayoutDocument`

The PageDisplayMode used for displaying the document. See also `TPageLayoutMode`.

7.1.9 IgnoreEmbeddedPreferences

Property (get, set): `bool IgnoreEmbeddedPreferences`
Default: `false`

When this property is set to `True` before opening a file, the viewer will ignore all viewing preferences that are embedded in said file. This includes open destination and the preferred layout mode.

7.1.10 PageNo

Property (get, set): `int PageNo`

The topmost pagenumber on the viewport.

7.1.11 PageOrder

Property (get, set): `List<int> PageOrder`
Default: `List<int>(Enumerable.Range(1, document.PageCount))`

Changing the viewing order of the displayed pages. The index of the list corresponds to the viewer page (viewer page 1 is at index 0, viewer page 2 at index 1 etc.) and the entry at that index corresponds to which PDF page will be displayed.

7.1.12 Zoom

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

The zoomFactor that the document is displayed with.

7.1.13 ShowOutlines

Property (get, set): `bool ShowOutlines`
Default: `true`

Determines if Outlines should be shown.

7.1.14 ShowThumbnails

Property (get, set): `bool ShowThumbnails`
Default: `true`

Determines if Thumbnails should be shown.

7.1.15 FileName

Property (get): `string FileName`
Default: `"-"`

The currently opened file name. File name is `"-"` if nothing is opened.

7.1.16 SearchOverlayBrush

Property (get, set): `System.Windows.Media.Brush SearchOverlayBrush`
Default: `Colors.LightBlue with Opacity=0.5`

Is the brush to use for the rectangular overlays of search results.

> **Windows Forms Note:** This property is implemented as a **System.Drawing.Color** with name **SearchOverlayColor**

7.1.17 MouseMode

Property (get, set): `TMouseMode MouseMode`
Default: `eMouseModeMove`

Sets the currently active mouse mode. See TMouseMode.

7.1.18 SearchMatchCase

Property (get, set): `bool SearchMatchCase`
Default: `true`

Configure, whether the search algorithm only finds matches with matching character case.

7.1.19 SearchWrap

Property (get, set): `bool SearchWrap`
Default: `true`

Configure, whether the search algorithm should wrap around when reaching the end of the document (or the start, if searching backwards).

7.1.20 SearchPrevious

Property (get, set): `bool SearchPrevious`
Default: `false`

Configure, whether the search algorithm searches for the next match (`False`, direction towards the end of the document) or the previous one (`True`, direction towards the frond of the document).

7.1.21 SearchRegex

Property (get, set): `bool SearchRegex`
Default: `true`

Configure, whether the search algorithm should interpret the given string as a regular expression (`True`), or a plain string (`False`).

7.1.22 ProductVersion

Property (get): `String ProductVersion`

Get the version of the 3-Heights™ PDF Viewer API R2 in the format "A.C.D.E".

7.1.23 ViewerPaneRectangle

Property (get): `System.Windows.Rect ViewerPaneRectangle`

The rectangle, where the viewer pane is situated within the WPF viewer Control in screen coordinates.

Windows Forms Note: This property is implemented as a **System.Drawing.Rectangle**

7.2 Methods

7.2.1 PdfViewerWPF

Method: `PdfViewerWPF()`

Constructor of Interfacing class PdfViewerWPF, instantiates an empty viewer, which can now be configured and used to open files.

> **Windows Forms Note:** This method is implemented with name **PdfViewer-WinForms**

7.2.2 Dispose

Method: `Dispose()`

Disposes Viewer and all its children. Must be called to terminate all running threads.

7.2.3 SetLicenseKey

Method: `SetLicenseKey(string licenseKey)`
Static

Set the key of your license.

7.2.4 GetLicenseIsValid

Method: `bool GetLicenseIsValid()`
Static

Returns whether the currently set key is valid.

7.2.5 InvokeCallbackOnDispatcher

Method: `InvokeCallbackOnDispatcher()`

Invoke the given callback on the Thread which owns this control. Returns after execution of callback terminates. All calls on the API should be performed on this thread.

7.2.6 BeginInvokeCallbackOnDispatcer

Method: `BeginInvokeCallbackOnDispatcer()`

Invoke the given callback on the Thread which owns this control. Returns immediately (not necessarily after termination of callback). All calls on the API should be performed on this thread.

7.2.7 Open

Method: `bool Open(string filename, string password)`

Instructs the viewer to open a file. Blocks until open operation has completed.

7.2.8 OpenMem

Method: `bool OpenMem(byte[] memBlock, string password)`

Instructs the viewer to open a file. Blocks until open operation has completed.

7.2.9 Search

Method: `Search(string searchText)`

Searches text of PDF for the inserted text. Search engine can be configured using Properties SearchWrap, SearchPrevious, SearchMatchCase and SearchRegex. Search results can be aquired by listening to the event SearchCompleted.

7.2.10 OnApplyTemplate

Method: `OnApplyTemplate()`

Initializes graphic components of the viewer. If PdfViewerWPF_R2 is used as a standard wpf component, this is called when tempaltes are applied. Otherwise manual invocation is necessary

> **Windows Forms Note:** This method is implemented seperately with name **Initialize**

7.2.11 SendInitialPropertyChanges

Method: `SendInitialPropertyChanges()`

Triggers DependencyPropertyChangedEvents of dependecyProperties LayoutMode, FitMode and PageDisplayMode, so Listeners to these events can read out initial values.

7.2.12 NextPage

Method: `NextPage()`

Set viewport to the next page.

7.2.13 PreviousPage

Method: `PreviousPage()`

Set viewport to the previous page.

7.2.14 SuspendLayout

Method: `SuspendLayout()`

Suspends Layouting the document until `ResumeLayout` is called.

7.2.15 ResumeLayout

Method: `ResumeLayout()`

Excplicitely forces layouting the document once and ends suspension caused by `SuspendLayout`.

7.3 Events

Java Swing API note: Events do not exist in older versions of Java. Thus there is an Interface defined for each event, containing a single method. One can implement said interface and write the code of the event handler in the given method. Then one has to register the implemented class on the viewer controller using the registerOn**PropertyName** name with the corresponding property name. E.g. the **OpenCompleted** event can be used by creating a class implementing **IOnOpenCompletedListener** and inserting the event handling code in the method **onOpenCompleted**. After registering an instance of the implemented class of the API using the method **registerOnOpenCompleted**, the **onOpenCompleted** method is then always called when a open operation terminates.
Also the **INotifyPropertyChanged** pattern and the associated **PropertyChanged** event does not exist in java. Instead there exists a separate event (i.e. the structure described above with an interface and a register method) for each property, notifying all registered listeners of the change in the properties value.

7.3.1 SearchCompleted

Event: `Action<PdfTools.PdfViewerCSharpAPI.Model.PdfSearcher.SearchResult> SearchCompleted`

Event triggers when method Search has been called and the search terminated. The event deliveres the result of the performed search as an Object of Type SearchResult, containing information about the found match:

- Pagenumber on which the match was found (if the match spans multiple pages, the first page will be returned)
- List of rectangles, that describe the location of the found match. (Long search terms yield multiple rectangles on multiple pages)
- Index describes the index of the first letter of the match within its page, to identify this match

7.3.2 TextExtracted

Event: `Action<String> TextExtracted`

Event triggers when text has been selected for extraction using the TMouseMode.eMouseTextExtractMode returning the selected text as one concatenated string.

7.3.3 OpenCompleted

Event: `Action<PdfTools.PdfviewerCSharpAPI.Utilities.PdfViewerException> OpenCompleted`

Event triggers when the opening of a file (e.g. by using Open) has completed. If there have been no errors while opening, the exception parameter will be null, otherwise it will carry the causing exception.

7.3.4 PageOrderChanged

Event: `Action<IList<int>> PageOrderChanged`

Event triggers when the PageOrder has been changed. It will return a list with the new page order.

7.3.5 ThumbnailLoaded

Event: `EventHandler<PdfViewerController.ThumbnailsChangedArgs> ThumbnailLoaded`

Event that triggers after a thumbnail has been loaded. `ThumbnailsChangedArgs` contains the bitmap, the page number of the thumbnail and an exception. The exception will be `null` if the loading of the thumbnail was successful.

7.3.6 PropertyChanged

Event: `System.ComponentModel.PropertyChangedEventHandler PropertyChanged`

Event triggers when any of the following properties changes its value: PageLayoutMode, FitMode, MouseMode, PageNo, PageCount, Zoom or Rotate. The event passes the string of the name of the property along.

8 Version History

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

8.1 Patches in Version 4.12

Note that the version number of the initial "final release" is 4.12.26.3.

Patch 4.12.26.4

- **Improved** error messages for failed HTTP connections in various situations (including license manager).
- **Added** missing documentation and release note for the proxy setting in the GUI license manager.
- **Improved** license reactivation behavior of the commandline license manager (licmgr): The server is now only contacted if necessary.
- **Improved** behavior of license manager when dealing with licenses of unreleased products.

8.2 Changes in Version 4.12

- **New** HTTP proxy setting in the GUI license manager.

8.3 Changes in Version 4.11

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

8.4 Changes in Version 4.10

- [Java] **New** method to print documents: `printDocument()`.
- [Java] **New** method `drawSinglePage()` to draw single pages into a `BufferedImage`
- [Java] **New** annotations:
  - Rectangle
  - Circle
  - Underline
- **Improved** robustness against corrupt input PDF documents.
- [C] **Clarified** Error handling of `TPdfStreamDescriptor` functions.

8.5 Changes in Version 4.9

- **Improved** support for and robustness against corrupt input PDF documents.
- **Improved** repair of embedded font programs that are corrupt.
- **New** support for OpenType font collections in installed font collection.
- **Improved** metadata generation for standard PDF properties.
- [C] **Changed** return value `pfGetLength` of `TPDFStreamDescriptor` to `pos_t`[4].

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

8.6 Changes in Version 4.8

- **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.

9 Tipps, Tricks and Troubleshooting

9.1 Troubleshooting: TypeInitializationException

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

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

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

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

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

Troubleshooting: DllNotFoundException

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

Resolve this by either:

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

Troubleshooting: BadImageFormatException

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

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

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

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

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