User Manual

3-Heights™ Java Document Viewer

PDF/A conform

Version 4.12.26.6

PDF-TOOLS.COM
Premium PDF Technology

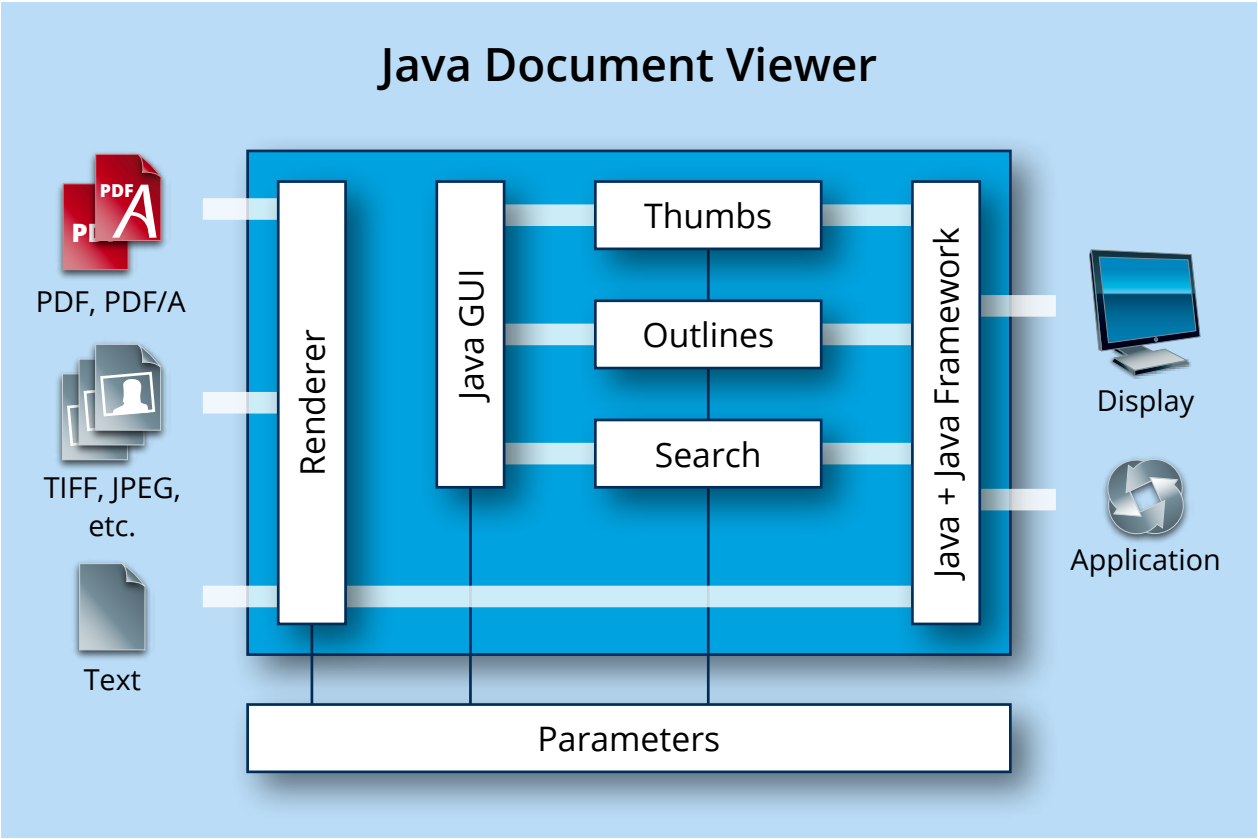
Contents

1 Introduction

1.1 Description

The 3-Heights™ Java Document Viewer is a quick and easy method of integrating flexible and customized viewer functionality in an application. Elements such as comments or highlights can be superimposed on the PDF, PDF/A, image and text files displayed on screen.

The user interface is freely designable and can be equipped with your own operating elements. This component is suitable for document management and enterprise content management systems, among others.



1.2 Functions

The 3-Heights™ Java Document Viewer can display and print images and documents and convert them into raster images. Pages are navigated by page number or the use of bookmarks or links. Displayed pages are freely scalable and visual comparisons between two documents are possible. Another tool is text extraction for full text search functionality. Extractable text can be displayed in a specific display mode. Elements such as comments or highlights from separate sources can be superimposed on the display. The Viewer also supports Asian writing systems, fill patterns, color gradients and transparencies.

1.3 Features

- Rendering of PDF documents (in accordance with the draft standard ISO 32000-1 (PDF 1.7))
- Rendering of PDF/A documents in accordance with ISO standard 19005-1
- Rendering of image files (TIFF, JPEG, BMP, JB2, JP2000, JPX, PNG, GIF)

- Rendering of text files (ASCII and Unicode UTF-16)
- Adaptable sample application as Java source code with customized menus, pushbuttons and much more
- Broad range of basic functions such as browse, scroll, zoom, rotate
- Page navigation by entering the page number or through the use of bookmarks or links
- Scaling (free scaling, no scaling, fit to page width, fit entire page)
- Display one or more pages, e.g. as miniature images (thumbnails)
- Display mode for displaying extractable text for use in digital signature applications
- Text extraction
- Text searching with regular expressions and high-lighting of multiple occurrences
- Visual comparison of two files
- Supports streaming interfaces for integrating components, for instance in archive systems
- Rendering of Chinese, Japanese and Korean writing systems (CJK)
- Rendering of fill patterns, color gradients and transparency

1.4 Formats

- PDF 1.x (PDF 1.0, . . ., PDF 1.7)
- PDF 2.0
- PDF/A-1, PDF/A-2, PDF/A-3
- Text files (ASCII, Unicode UTF-16)
- BMP (1, 2, 4, 8, 24 bit)
- GIF (2 to 8 bit)
- JBIG2 (lossy compression, lossless compression)
- JPEG (Grayscale, RGB, CMYK)
- JPEG2000 and JPEG-LS (Grayscale, RGB, CMYK)
- PBM (1 to 8, 24 bit)
- PNG (1 to 8, 24 bit)
- TIFF
  - Bitonal: uncompressed, CCITT G3, CCITT G3-2D, CCITT G4, LZW, ZIP, Packbits
  - Grayscale: uncompressed, LZW, JPEG, JPEG (old), ZIP, Packbits
  - RGB: uncompressed, LZW, JPEG, JPEG (old), ZIP, Packbits
  - CMYK: uncompressed, LZW, JPEG, JPEG (old), ZIP, Packbits

1.5 Compliance

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

1.6 Operating Systems

The 3-Heights™ Java Document Viewer is available for the following operating systems:

- Windows 7, 8, 8.1, 10 – 32 and 64 bit
- Windows Server 2008, 2008 R2, 2012, 2012 R2, 2016 – 32 and 64 bit
- HP-UX 11i and later PA-RISC2.0 – 32 bit
- HP-UX 11i and later ia64 (Itanium) – 64 bit
- IBM AIX 6.1 and later – 64 bit

- Linux 2.6 – 32 and 64 bit
- Oracle Solaris 2.8 and later, SPARC and Intel
- FreeBSD 4.7 and later (32 bit) or FreeBSD 9.3 and later (64 bit, on request)
- macOS 10.4 and later – 32 and 64 bit

2 Installation

2.1 Windows

- Download the ZIP archive of the product from your download account at www.pdf-tools.com.
- Open the ZIP archive.
- Check the appropriate option to preserve file paths (folder names) and unzip the archive to a local folder (e.g. C:\program files\pdf-tools\).
- The unzip process now creates the following subdirectories and files:

bin\‹platform›\PDFViewerAPI.dll	This DLL contains the main functionality (required).
doc*	Contains documentation files.
jar\VWJA.jar	Java API archive.
samples	Sample applications.

2.1.1 Installation and Running on the CMD Shell

There are different ways how to compile and run a Java application, here is one:

1. Set the environment variable CLASSPATH to include the Jar file VWJA.jar. (Note: do not use blanks before and after the equal sign.)

```
set CLASSPATH=%CLASSPATH%;C:\pdf-tools\JavaViewer\jar\VWJA.jar
```

or set it new, ignoring the old setting:

```
set CLASSPATH=.;C:\pdf-tools\JavaViewer\jar\VWJA.jar
```

2. Compile the application.

```
javac PdfViewerApp.java
```

3. Adjust the environment variable PATH to include the directory where the file PdfViewerAPI.dll resides.

```
set PATH=%PATH%;C:\pdf-tools\JavaViewer\bin\‹platform›
```

4. Run the application.

```
javaw PdfViewerApp
```

Optionally with a file name as parameter:

```
javaw PdfViewerApp C:\pdf\somefile.pdf
```

2.1.2 Troubleshooting

Message:

```
Exception in thread "main" java.lang.UnsatisfiedLinkError: no PdfViewerAPI in
  java.library.path
```

Problem The file `PdfViewerAPI.dll` cannot be found.

Solution Add the directory where it resides to the PATH.

Message:

```
PdfViewerApp.java:29: package com.pdftools.pdfviewer does not exist
```

Problem The `vwja.jar` file cannot be found.

Solution Add it to the CLASSPATH.

Message:

```
Exception in thread "main" java.lang.ExceptionInInitializerError
Caused by: java.lang.Exception: The library version 1.6.0.37 doesn't match
  with the java interface version 1.6.0.36.
```

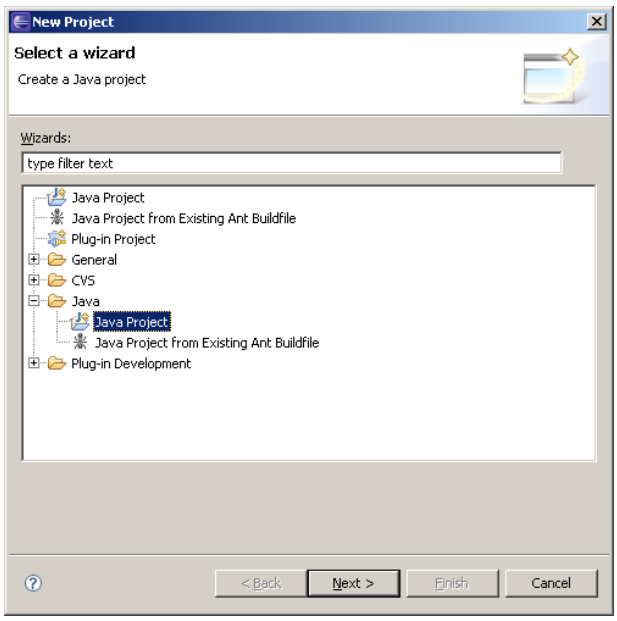
Problem The `vwja.jar` and `PdfViewerAPI.dll` are of two different versions.

Solution Use files from the same version and make sure you are referring to those when running the application.

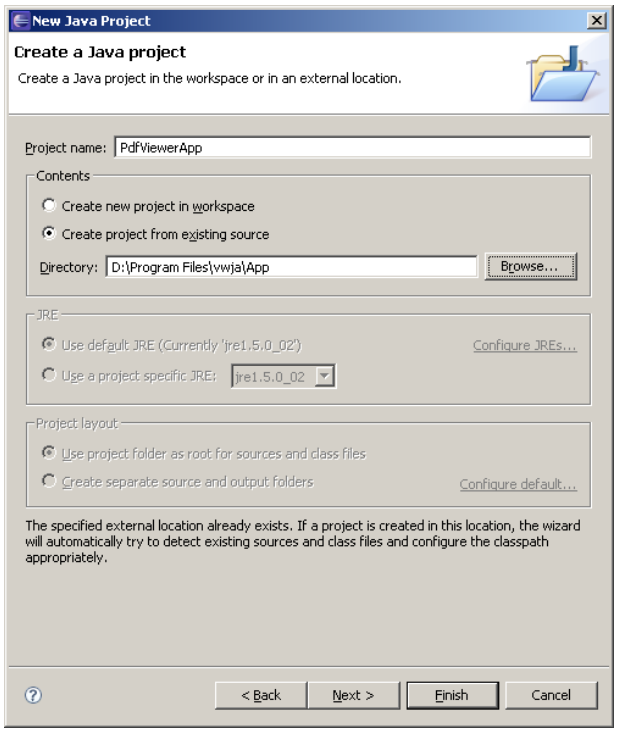
2.2 Eclipse

(The following steps are described for the Eclipse 3.2.)

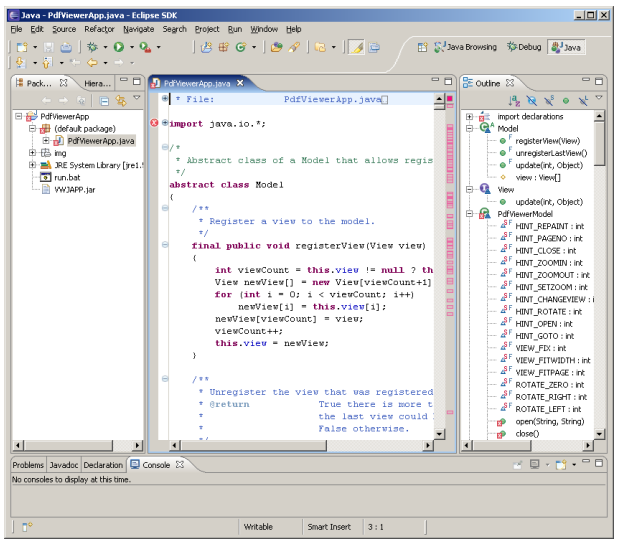
1. Start Eclipse.
2. Create a new Project by selecting “File” →“New” →“Project...”.
3. Select item “Java Project”.
4. Press button “Next”.



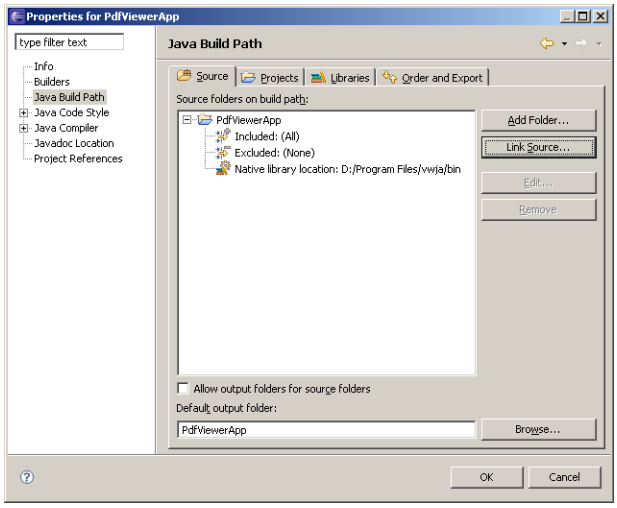
5. Name the project “PdfViewerApp”.
6. Select the radio button “Create project from existing source”.
7. Set the directory to where the file `PdfViewerApp.java` resides.
8. Press button “Finish”.



9. A project is now created, that should look similar as in the screenshot to the right.
10. Right-click the project “PdfViewerApp” and select properties.

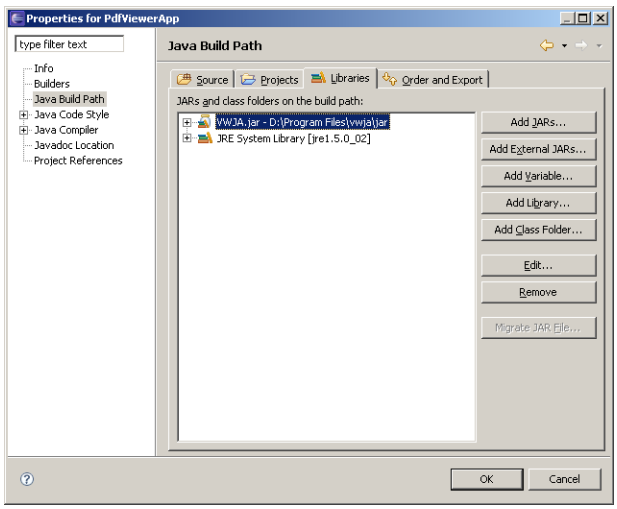


11. On the left side select “Java Build Path”.
12. On the right side select the tab “Source”.
13. Set the native library location to where the `PDFViewerAPI.dll`[1] resides.

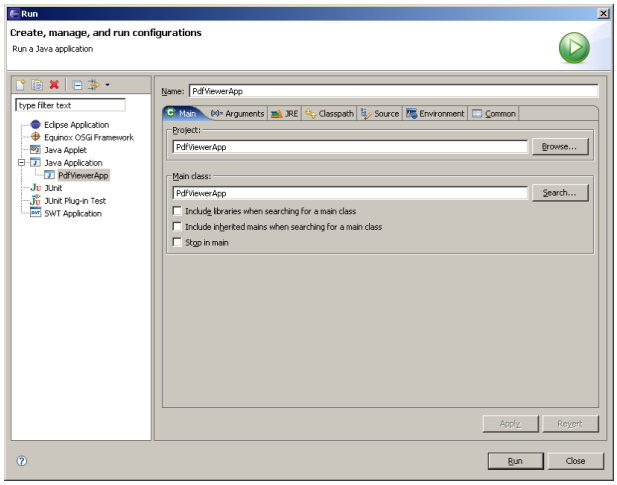


14. On the right side select the tab “Libraries”.
15. Add the Jar file `VWJA.jar`.

[1] In Unix this is called `libPdfViewAPI.so`, in macOS `libPdfViewerAPI.dylib`.



16. Optional: Set the JDK Compliance level to 1.6.
17. From the menu “Run” select “Run. . .”.
18. Name the configuration (e.g. “PDFViewerApp”).
19. Select the project “PDFViewerApp”.
20. Select the main class “PDFViewerApp”.
21. Press button “Run”.
22. We are done.



2.3 Unix

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

Here is an overview of the files that come with the 3-Heights™ Java Document Viewer:

File Description

Name	Description

File	Description
bin/‹platform›/libPdfViewerAPI.so	This is the shared library that contains the main functionality. The file's extension varies depending on the type of UNIX system. The directory ‹platform› is either x86 containing the 32-bit version of the library, or x64 for the 64-bit version.
doc/*.*	Documentation
jar/VWJA.jar	Java API archive.
samples	Example code.

2.3.1 All Unix Platforms

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

     ```
     ldd libPdfViewerAPI.so
     ```

   - *On AIX*:

     ```
     dump -H libPdfViewerAPI.so
     ```

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

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

4. Optionally register your license key using the Command Line License Manager Tool.
5. Add the directory containing libPDFViewerAPI.dylib to the DYLD_LIBRARY_PATH.
6. Add the VWJA.jar file to the CLASSPATH.

2.3.2 macOS

The shared library must have the extension .jnilib for use with Java. We suggest that you create a file link for this purpose by using the following command:

```
ln libPdfViewerAPI.dylib libPdfViewerAPI.jnilib
```

2.4 Note about the Evaluation License

With the evaluation license the 3-Heights™ Java Document Viewer automatically adds a watermark to the displayed pages.

3 Design Principles

1. The native code contains the implementation of the graphics engine independent objects (`PdfViewer-API.dll`).
   a. For performance reasons as much as possible, e.g. converting colors, is done in the native code.
   b. The contents of a page are represented by two types of objects: resources and operators which are created by the native code.
2. The Graphcis2D specific code is concentrated in the compiled Java code (`VWJA.jar`).
   a. The interface design follows a suitable subset of the PDF semantics; the subset is chosen such that the objects can easily be used by Graphics2D, e.g. only sRGB colors are used.
   b. The interface to the native code consists of a few, easy-to-use classes:
      I. PdfDocument: Opens a document and provides access to its pages.
      II. PdfPage: Provides access to the page's attributes and operator stream.
      III. PdfTextExtractor: Returns the text contained in a page.
      IV. PdfTextFinder: Finds the text contained in a page and matching a regular pattern.
      V. PdfPageFinder: Finds all pages containing text matching a regular pattern.
3. The user interface is provided as Java source code (`AWTViewerAdvanced`, `AWTViewerSimple`). It contains the code for the following features:
   a. Display a pages
   b. Links and go to link destination
   c. Scroll, zoom, fit modes and rotate
   d. Navigate: first, next, previous, last page, go to page
   e. Keyboard navigation
   f. Mouse scroll, zoom, move and activate link
   g. Bookmark tree
   h. Page thumbnails
   i. Find text, highlight and go to next occurrence
4. Some other applications of the component such as printing and conversion of a PDF to an image are provided as source code as well (`AWTConverter`, `AWTPrinter`).

4 License Management

The 3-Heights™ Java Document Viewer requires a valid license in order to run correctly. If no license key is set or the license is not valid, then the 3-Heights™ Java Document Viewer will fail.

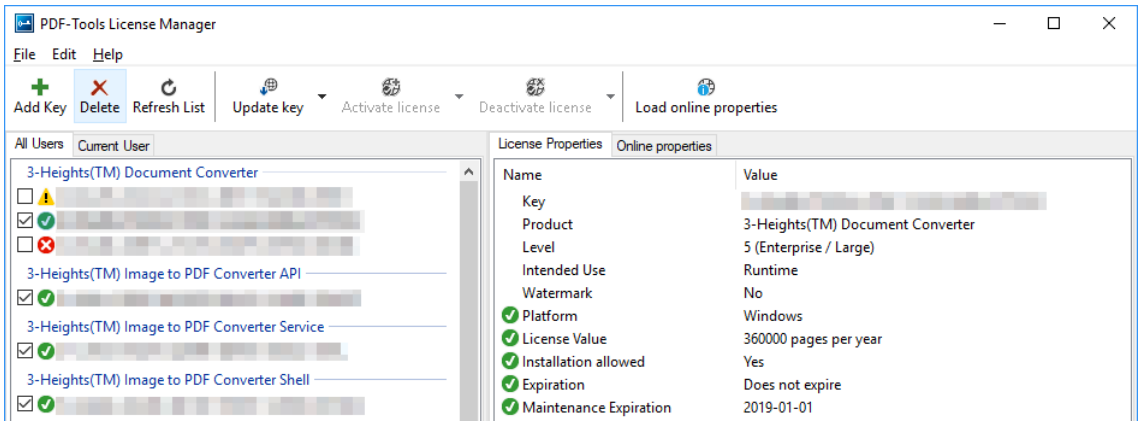
4.1 License Installation and Management

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.

4.1.1 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.

4.1.2 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
```

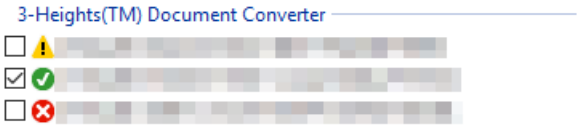
4.2 License Selection and Precedence

4.2.1 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
```

4.2.2 Precedence

License keys are considered in the following order:

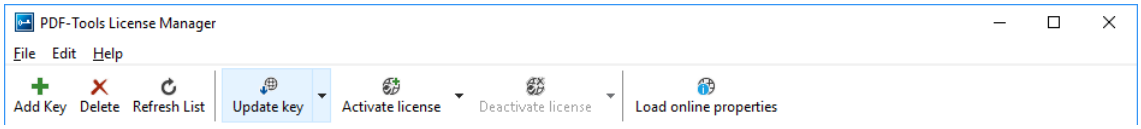
1. License key passed at runtime.
2. License selected for the current user
3. License selected for the current user (legacy key format)
4. License selected for all users
5. License selected for all users (legacy key format)

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

4.3 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
```

4.4 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.

4.4.1 Activation

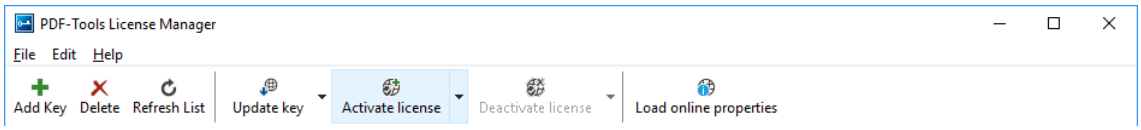
The License can be activated directly within the license manager. Every activation increases the activation count of the license by 1.

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:

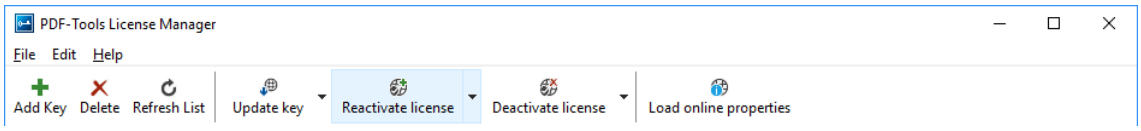
```
licmgr activate -cd 1-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
licmgr activate -c "custom comment" 1-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

4.4.2 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:

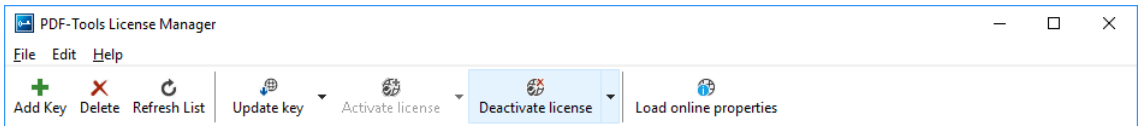
```
licmgr activate 1-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

4.4.3 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:

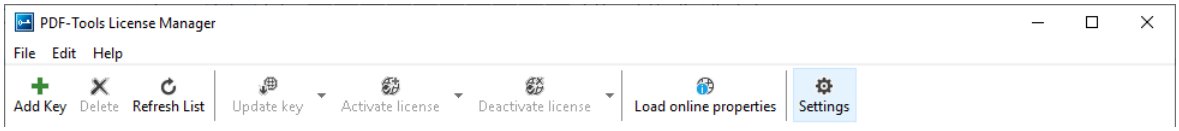
```
licmgr deactivate 1-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
```

4.5 Proxy Setting

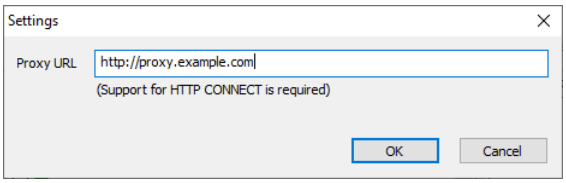
A proxy URL can be configured for computers that cannot access the internet without a web proxy.

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:



4.6 Offline Usage

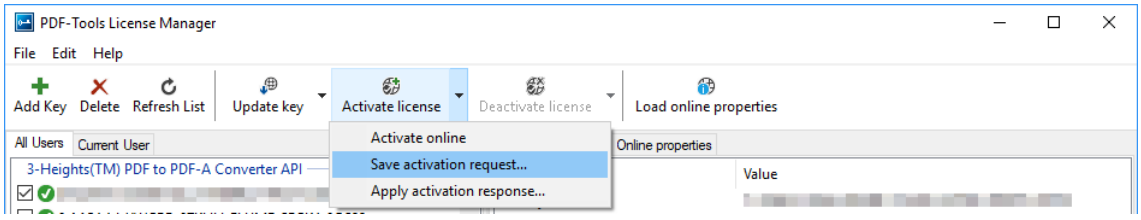
The following actions in the license manager need access to the internet:

- 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.

4.6.1 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:



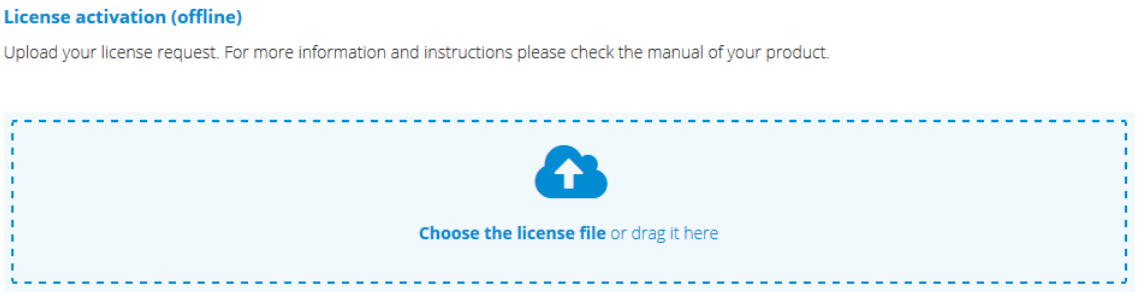
With the Command Line Interface use the `-fs` option to specify the destination path of the request file:

```
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.

4.6.2 Second Step: Use Form on Website

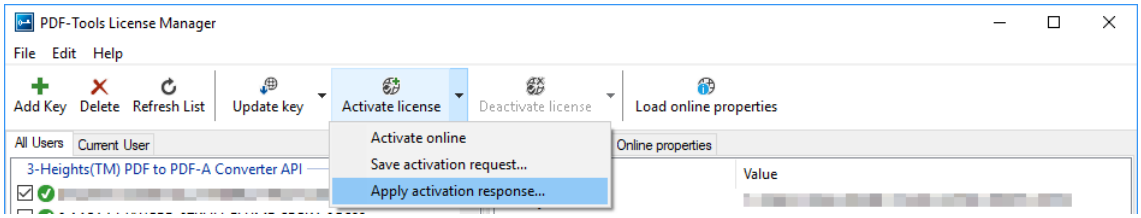
Open the following website in a web browser: http://www.pdf-tools.com/pdf20/en/mypdftools/licenses-kits/license-activation/ Upload the request by dragging it onto the marked area:



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

4.6.3 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
```

4.7 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.

4.8 License Key Storage

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

4.8.1 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)

4.8.2 macOS

The license keys are stored in the file system:

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

4.8.3 Unix/Linux

The license keys are stored in the file system:

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

> **Note:** The user, group and permissions of those directories are set solely by the license manager tool. It may be necessary to change permissions to make the licenses readable for all users. Example:
>
> ```
> chmod -R go+rx /etc/opt/pdf-tools
> ```

4.9 Troubleshooting

4.9.1 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.

4.9.2 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.

4.9.3 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.

4.9.4 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.

4.9.5 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.

4.9.6 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.

4.9.7 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.

4.9.8 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 4.6 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.

5 User's Guide

There is an extensive GUI application called `PDFViewerApp.java` included in the 3-Heights™ Java Document Viewer (Release and Evaluation Versions) that uses Java Swing. It shows how to display pages in certain modes like fit-page, fit-width or original size, how to scroll, zoom, rotate a page or create an outline (bookmark) tree, etc. The idea of this chapter is to understand how this application is built and how change can be applied or enhancements can be added.

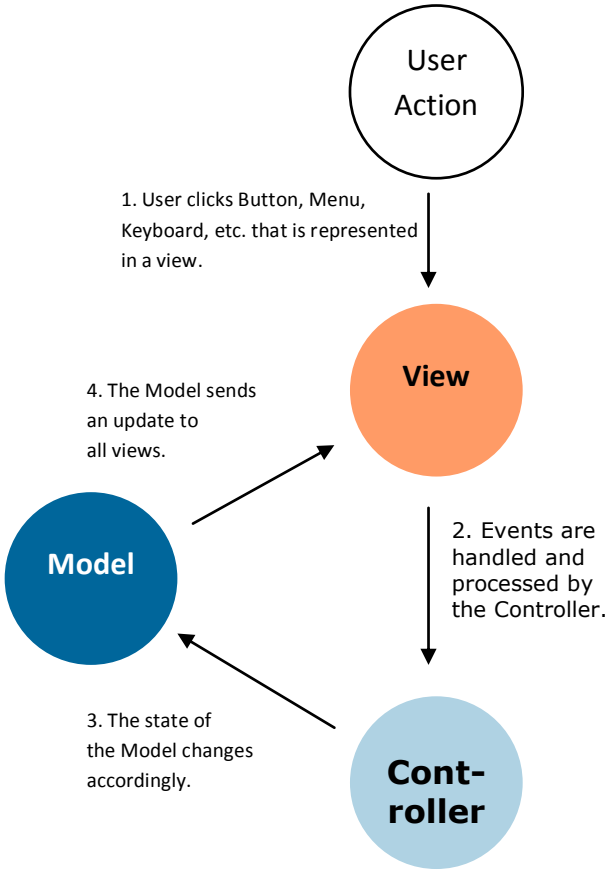
> **Note:** Code snippets in this manual might differ from the actual code, due to modifications or enhancements.

5.1 Model-View Design

The idea of introducing a model and a view is to divide the code into the common Swing architecture that consists of a Model, a View and a Controller. There are many articles about the so called MCV architecture, one can be found at http://www.oracle.com/technetwork/java/architecture-142923.html.

Here is a brief overview:

`The Model`	The model represents the data of the application. It keeps track of which file currently is open, how many pages there are, and the current page.
`The View`	The visual representation of the data. It displays the actual page. There may be multiple views, for example a main view and a thumbnail view. Different views can potentially display different pages at different zoom levels and with different rotations. Therefore view specific information is saved in the view itself.
`The Controller`	The controller handles interactions with the user. Inputs from views are processed and forwarded to update the model.



5.1.1 The Model

The model must inform all views about changes in the model. E.g. if the document is closed and another document is opened, the data in the model changes and all views must be informed and react adequately, e.g. display the new document.

The abstract class `Model` provides the functionality to register views. All views that are registered in the model are sent an update if the model has been updated.

The derived class `PdfViewerModel` contains a `PdfViewer` and the current page number as member variables. The class `PdfViewer` contains information about the actual document, such as the filename or the number of total pages. It provides functions to `open` and `close` a document as well as to go to a certain page number or changing the way how to display a page. Static hints that are provided with the update function give additional information what and how the view must update.

```
abstract class Model
{
    // View Registering/Update
    final public void registerView(View view)
    final public boolean unregisterLastView()
    final public void update(int hint, Object param)
    // Member Variables
    protected View view[];
}
```

```
class PdfViewerModel extends Model
{
```

```
    // Document Open/Close
    public boolean open(String fileName, String password)
    public void close()
    // Goto Page
    public void goFirstPage()
    public void goLastPage()
    public void goNextPage()
    public void goPreviousPage()
    public void goPage(int pageNo)
    public void goTo(int pageNo, float x, float y, float z)
    // Zooming/Scaling/Rotating
    public void zoomIn()
    public void zoomOut()
    public void setZoom(float Zoom)
    public void setActualSize()
    public void setFitWidth()
    public void setFitPage()
    public void rotateRight()
    public void rotateLeft()
    public void rotateZero()
    // TextMode
    public void setTextMode(boolean bsetTextMode)
    // Member Variables
    protected PdfViewer viewer;
    protected int pageNo;
    protected Rectangle visibleRect;
}
```

5.1.2 The View

The view visualizes the data. For a Java Document Viewer, the most obvious way how to achieve this is to draw the page content. The class `PdfViewerPageView` does exactly that. Its `update` function reacts to commands sent by the model. It contains a paint function that can actually draw the current page when needed. It contains several member variables, the three most important are: The current `pageNo` as integer, the current `page` object and a `PdfViewerApp`. You might ask why the page number has to be stored in the view as well as in the model. There are different reasons for that. The page number in the model represents what is being considered the current page, i.e. the page number that is displayed in the main view. Different views however might display different page numbers, the main view can display page 1, whereas the thumbnail view might currently display pages 4 to 8. When the user selects a new page number using for example the thumbnail view interface, the page number of the model changes, and as a result the page number in the views are updated as well.

There is also an outline (bookmark) and list (page list) which provide textual views of the pages.

```
interface View
{
    abstract public void update(int hint, Object param);
}
```

```
class PdfViewerPageView extends JComponent implements View, ActionListener,
    MouseListener
{
    // Constructor
    PdfViewerPageView(PdfViewerApp app)
    // Painting
```

```
    private void paintDescription(String description, Point pt)
    private void paintMarkRect()
    public Dimension getPreferredSize()
    public void paint(Graphics g)
    // Class Specific/Helper
    private PdfDestination getDestAt(MouseEvent e)
    private AffineTransform getTrans(boolean isInverted)
    // Events/Update
    public void mouseClicked(MouseEvent e)
    public void mouseEntered ...
    public void actionPerformed(ActionEvent a)
    public void update(int hint, Object param)
    // Member Variables
    protected final PdfViewerApp app;
    protected PdfViewerPage page;
    protected int pageNo;
    protected float zoomFactor;
    protected int pageView;
    protected int viewingRot;
    private final float dpiFactor;
    private Point p1, p2;
    private PdfAnnotation links[];
    private Rectangle rectPrev, rectCur;
    private int curserMode;
}
```

```
class PdfViewerThumbnailView extends Component implements View, MouseListener
{
    // Painting
    public Dimension getPreferredSize()
    public void paint(Graphics g)
    // Events/Update
    public void mouseClicked(MouseEvent e)
    public void mouseEntered ...
    public void update(int hint, Object param)
    // Member Variables
    protected PdfViewerPage page;
    protected final int thumbSize;
    protected final PdfViewerApp app;
}
```

```
abstract class PdfTreeView extends JPanel implements View
{
    // Constructor
    PdfTreeView(PdfViewerApp app)
    // Tree
    abstract protected void buildTree();
    protected void expand(JTree tree, DefaultMutableTreeNode node)
    // Update
    public void update(int hint, Object param)
    public Dimension getPreferredSize()
    // Member Variables
    protected JTree tree;
    protected final PdfViewerApp app;
```

```
}
```

```
class PdfOutlineView extends PdfTreeView
{
    // Constructor
    PdfOutlineView(PdfViewerApp app)
    // Tree
    protected void buildTree()
    // Member Variables
    protected JTree treeExp;
}
```

```
class PdfListView extends PdfTreeView
{
    // Constructor
    PdfListView(PdfViewerApp app)
    // Tree
    protected void buildTree()
}
```

5.1.3 The Controller

The controller launches the application. It sets up the GUI and processes user actions in its function `actionPerformed`.

```
// Main Class
public final class PdfViewerApp extends JFrame implements View, ActionListener
{
    // Constructor
    public PdfViewerApp(String fileName, String password)
    // Main
    public static void main(String args[])
    // GUI Builders
    public void showView(int showViewNew)
    private final void makeMenu()
    private final void addMenuFile(JMenuBar menuBar)
    private final void addMenuDocument(JMenuBar menuBar)
    private final void addMenuView(JMenuBar menuBar)
    private final void makePanel()
    // Class Specific/Helper
    public boolean isDisableLink()
    private ImageIcon loadIcon(String name)
    // Events/Update
    public void update(int hint, Object param)
    public void actionPerformed(ActionEvent e)
    public Dimension getPreferredSize()
    // Member Variables
    protected static int showView;
    protected JScrollPane thumbPane, outlinePane, listPane;
    protected final PdfViewerModel model;
    protected String fileName;
}
```

```
// Helper Classes
class PdfViewerScrollPane extends JScrollPane
{
    PdfViewerScrollPane(PdfViewerPage View component)
    protected final PdfViewerPageView view;
}

class JTextFieldZoom extends JTextField
{
    JTextFieldZoom(int columns, final ActionListener listener)
}
```

5.2 Setting Up the GUI

The graphics user interface (GUI) is setup in the constructor of the `PdfViewerApp`. The relevant steps done are as following:

1. Input parameters are parsed for in the `main` function and a file name and password is set if provided.
2. The main function instantiates a `PdfViewerApp` which is derived from `JFrame`.
3. A model is instantiated. Any view will be registered to the model and be notified by the model using the `update` function.
4. The `PdfViewerApp` implements a `View` itself in (order to update the frame title if a new document is opened). It is therefore registered in the model.
5. In the next step a menu bar and its menu icons are created.
6. A Panel with buttons is added.
7. The main-window-view `PdfViewerPageView` is instantiated and registered in the model.
8. A derived `ScrollPane` is instantiated and added the `PdfViewerPageView`. The scroll pane is added to the content pane of the frame.
9. A document is opened if initially provided.
10. The frame is displayed.

```
public PdfViewerApp(String fileName, String password)
{
    model = new PdfViewerModel();
    model.registerView(this);

    makeMenu();
    makePanel();

    PdfViewerPageView mainView = new PdfViewerPageView(this);
    model.registerView(mainView);

    PdfViewerScrollPane viewerPane = new PdfViewerScrollPane(mainView);
    getContentPane().add(viewerPane, "Center");
    viewerPane.setMinimumSize(new Dimension(10, 10));

    setTitle("PDF Viewer");
    if (fileName.length() > 0)
    {
        this.fileName = fileName;
        model.open(fileName, password);
    }
    showView(PdfViewerApp.showOutline);
}
```

```
public static void main(String args[])
{
    String fileName = "";
    String password = "";
1   if (args.length >= 1)
        fileName = args[0];
    if (args.length >= 2)
        password = args[1];
    showView = PdfViewerApp.showNone;
2   PdfViewerApp app =
        new PdfViewerApp(fileName, password);
10  app.pack();
    app.setResizable(true); app.setVisible(true);
    app.setDefaultCloseOperation(Window Constants.EXIT_ON_CLOSE);
}
```

5.3 Menubars and Buttons

How to add menu bars and buttons is not really related to the PDF Java Viewer, however they are required to provide the user the possibility to interact with the application. Most functionalities require an interaction, therefore this is described first.

1. Menu and Panel related objects (`JMenuItem`, `JMenu`, `JCheckBoxMenuItem`, `JLabel`, etc.) are defined in the Controller.
2. The Menu and Panel are created in functions that are called by the constructor of `PdfViewerApp`.
3. `makeMenu` calls three sub-functions. Each of those creates one menu including all menu items. Also key shortcuts are added.
4. `makePanel` creates a panel and adds buttons and labels onto it. Most buttons use an image which is imported using the helper function `loadIcon`.

```
public final class PdfViewerApp extends JFrame implements View, ActionListener
{
1   private JMenuItem menuItem_Open, menuItem_Close, ...
    private JMenu menu_ZoomTo, menu_Navigation;
    private JCheckBoxMenuItem, checkBoxMenuItem_TextMode, ...

    public PdfViewerApp(String fileName, String password)
    {
        ...
2       makeMenu();
        makePanel();
        ...
    }

3   private final void makeMenu()
    {
        JMenuBar menuBar = new JMenuBar();
        addMenuFile(menuBar);
        addMenuDocument(menuBar);
        addMenuView(menuBar);

        setJMenuBar(menuBar);
    }

    private final void addMenuFile(JMenuBar menuBar)
```

```
    {
        JMenu menu = new JMenu("File");
        menu.setMnemonic(KeyEvent.VK_F);
        menuBar.add(menu);

        menuItem_Open = new JMenuItem("Open", KeyEvent.VK_O);
        menuItem_Open.setAccelerator( KeyStroke.getKeyStroke(KeyEvent.VK_O,
          ActionEvent.ALT_MASK));
        menuItem_Open.addActionListener(this);
        menu.add(menuItem_Open);
        menuItem_Close = new JMenuItem("Close", KeyEvent.VK_C);
        ...
    }

    private final void addMenuDocument(JMenuBar menuBar)

    private final void addMenuView(JMenuBar menuBar)

    private ImageIcon loadIcon(String name)
    {
        return new ImageIcon(PdfViewerApp.class.getResource("gfx/" + name));
    }

    private final void makePanel()
    {
        // Buttons
        JPanel panel = new JPanel();
        panel.setLayout(new FlowLayout(FlowLayout.LEFT));
        button_Open = new JButton(loadIcon("open.gif"));
        button_Open.addActionListener(this);
        button_First = new JButton(loadIcon("first.gif"));
        button_First.addActionListener(this);
        ...

        // Page number
        ...

        getContentPane().add(panel, "North");
    }
}
```

5.4 Handling Events

Events (open file, go to page, etc.) triggered by a menu or a button are caught by the function `actionPerformed` in the controller. An event is processed and results in a change of the state of the model.

```
public void actionPerformed(ActionEvent e)
{
    if (e.getSource() == menuItem_Open || e.getSource() == button_Open)
    {
        final JFileChooser fc = new JFileChooser();
        if (fc.showOpenDialog(this) == JFileChooser.APPROVE_OPTION)
        {
            final String filename = fc.getSelectedFile().getAbsolutePath();
            if (model.open(filename, ""))
                model.setTextMode(checkBox MenuItem_ TextMode.getState());
```

```
            this.repaint();
        }
    }
    else if (e.getSource() == menuItem_Close)
        model.close();
    ...
}
```

5.5 Open and Close a Document

The currently opened document is stored in the model.

1. Closing a document or opening a new document is handled by the model.
2. A call to `update` informs all views. A hint given as parameter in the `update` function provides additional information, i.e. what exactly happened (document closed, document opened, etc.).
3. Each active view handles the updates individually. For a view it is not required to address all update calls. The main view (`PdfViewerPageView`) for example handles the close-hint, but not the open-hint. For this view it is only relevant when something happens to the active page. This is the case when no page needs to be drawn (close), however it does not happen with the open-hint, but with the subsequent pageno-hint
4. which results as a consequence of `goFirstPage`.

```
class PdfViewerModel extends Model
{
    final static int HINT_CLOSE = 4;
    final static int HINT_OPEN = 256;

1   public boolean open(String fileName,
    String password) {
        if (viewer != null)
           close();
        try
        {
             viewer = new PdfViewer(fileName, password);
2            update(PdfViewerModel.HINT_OPEN, fileName); setZoom(1.0f);
4            goFirstPage();
        }
        catch (FileNotFoundException e)
        {
             return false;
        }
        return true;
    }

    public void close()
    {
        if (viewer != null)
        {
             pageNo = 0;
2            update(HINT_CLOSE,
                 new Integer(0));
             viewer.destroyObject();
             viewer = null;
        }
    }
}
```

```
class PdfViewerPageView extends JComponent implements View, ActionListener,
    MouseListener
{
3   public void update(int hint, Object param)
    {
        if ((hint & PdfViewerModel.HINT_CLOSE) != 0)
        {
            if (page != null)
            {
                page.destroyObject();
                page = null;
                pageNo = 0;
                viewingRot = 0;
                repaint();
                setSize(getParent().getSize());
            }
        }
        else if ((hint & PdfViewerModel.HINT_PAGENO) != 0)
        {
            final int pageNo = ((Integer) param).intValue();
            if (pageNo < 1 || pageNo > app.model.viewer.getPageCount())
                return;

            if (this.pageNo != pageNo)
            {
                this.pageNo = pageNo;
                if (page != null)
                    page.destroyObject();
                page = app.model.viewer.getPage(pageNo);
                viewingRot = page.getRotate();
                repaint();
            }
        }
    }
}
```

```
class PdfViewerThumbnailView extends Component implements View, MouseListener
{
3   public void update(int hint, Object param)
    {
        if ((hint & PdfViewerModel.HINT_OPEN) != 0)
        ...

        else if ((hint & PdfViewerModel.HINT_CLOSE) != 0)
        ...
    }
}
```

5.6 Draw a Page

Drawing a page is done using the `paint` function if the view.

1. In a first step the currently viewed window of the page must be determined. This is done using the function `getSize`. `getSize` returns the size of the page in form of a dimension object. The size was previously set using `setSize` and depends on the current zoom factor.
2. The content of the page is drawn into a given viewport container using a Graphics 2D container. What is a viewport? The viewport is the region of a frame to be displayed. Content outside the viewport is not displayed.
3. If no page is active, a blue background rectangle is drawn instead.

```
class PdfViewerPageView extends JComponent implements View, ActionListener,
    MouseListener
{
    public void paint(Graphics g)
    {
1       final Dimension size = getSize();
        if (page != null)
        {
            final Rectangle rect = new Rectangle(0, 0, size.width,
                size.height);
2           page.draw((Graphics2D) g, rect);
            ...
        }
3       else
        {
            g.setColor(new Color(174, 209, 226));
            g.fillRect(0, 0, size.width, size.height);
        }
    }
}
```

5.7 Rotation

When it comes to rotation, it needs to be considered that each page in a PDF document may have an individual viewing rotation. Often the viewing rotation attribute is abused to display a rotated portrait page as landscape. The initial viewing rotation and the user rotation define the resulting rotation. Here is how it works:

1. The User triggers an event which is caught by the controller. The controller calls the corresponding function in the model, e.g. `rotateRight`.
2. `rotateRight` updates all views by sending a rotate-hint.
3. All views, which should rotate pages, have the appropriate hint handled by their update function. Rotation is usually only handled by the main view, the thumbnail view should rather stay not rotated.
4. The view rotates the page by the selected value, the initial rotation (`viewingRot`) and previous rotations (`page.getRotate`) need to be considered.
5. The page needs to be repainted.

```
class PdfViewerModel extends Model
{
    final static int HINT_ROTATE = 128;
    final static int ROTATE_ZERO = 0; // Set rotation to 0
    final static int ROTATE_RIGHT = 1; // Rotate by 90 degrees
    final static int ROTATE_LEFT = 2; // Rotate by -90 degrees
1   public void rotateRight()
```

```
    {
        if (viewer != null)
2           update(HINT_ROTATE, new Integer(ROTATE_RIGHT));
    }

    public void rotateLeft()

    public void rotateZero()
}
```

```
class PdfViewerPageView extends JComponent implements View, ActionListener,
    MouseListener
{
3   public void update(int hint, Object param)
    {
        ...
        else if ((hint & PdfViewerModel.HINT_ROTATE) != 0)
        {
            final int rotate = ((Integer) param).intValue();
            if (rotate == PdfViewerModel.ROTATE_RIGHT)
4               page.setRotate(page.getRotate() - viewingRot + 90);
            else
                ...
            setSize(getPreferredSize());
5           repaint();
        }
    }
    protected int viewingRot;
}
```

When opening a new document, the rotation is reset as one can see in the open and update parts of Open and Close a Document.

5.8 Go to Page

1. The model provides some high level function that jumps directly to a certain page number (first, last, next, previous).
2. They are all interpreted to end up in a call to goPage, which sets an actual page number.
3. goPage verifies the page is within a valid range and sends an update to all views. (For the update of the pageno-hint, see chapter Open and Close a Document.)

```
class PdfViewerModel extends Model
{
1   public void goFirstPage()
    {
        if (viewer != null)
        {
            pageNo = 1;
            goPage(pageNo);
        }
    }
    public void goLastPage()

    public void goNextPage()
```

```
    public void goPreviousPage()

    public void goPage(int pageNo)
    {
        if (viewer != null)
        {
            if (pageNo < 1)
                pageNo = 1;
            else if (pageNo > viewer.getPageCount())
                pageNo = viewer.getPageCount();
            this.pageNo = pageNo;
            update(HINT_PAGENO, new Integer(pageNo));
        }
    }
}
```

5.9 Fit Modes

The PDF Java Viewer GUI application supports three viewing mode:

- Fixed: Means the zoom factor does not depend on the frame size, the content is displayed with a defined size (e.g. actual size).
- Fit width: Means the zoom factor is set so that page width is always adjusted to that it exactly fits the width of the frame.
- Fit page: Means to zoom factor is set so that the full page is always visible.

These fit modes are pure GUI functions that take into account the size of the current page and the size of the view port. There is no fit mode function in the API. The function `getPreferredSize` is usually called before a call to `repaint`. It retrieved the size of the current page and the current view port (depending on the size of the frame). It calculates the zoom factor and returns the dimensions of the page that needs to be drawn.

1. The GUI applications draws the page onto a pane. In the fixed mode, the pane can have any size. The larger the pane, the smaller the relative part that can be displayed given a fixed viewport size and using a `JScrollPane`. In this case the zoom factor is given by the user and does not need to be calculated. `getPreferredSize` returns the dimension of the page at the given zoom factor multiplied by a scaling dpiFactor, to take into account a PDF has a resolution of 72 DPI (dots per inch) where as a monitor has usually a resolution of 96 DPI.
2. In the fit-width mode, the page must always fit within the given width of the pane. Therefore the zoom factor needs to be re-calculated. The dimension that is returned has the appropriate size to paint the page at the given zoom factor so that its width fits in.
3. For the fit-page mode, both width and height have to be considered.

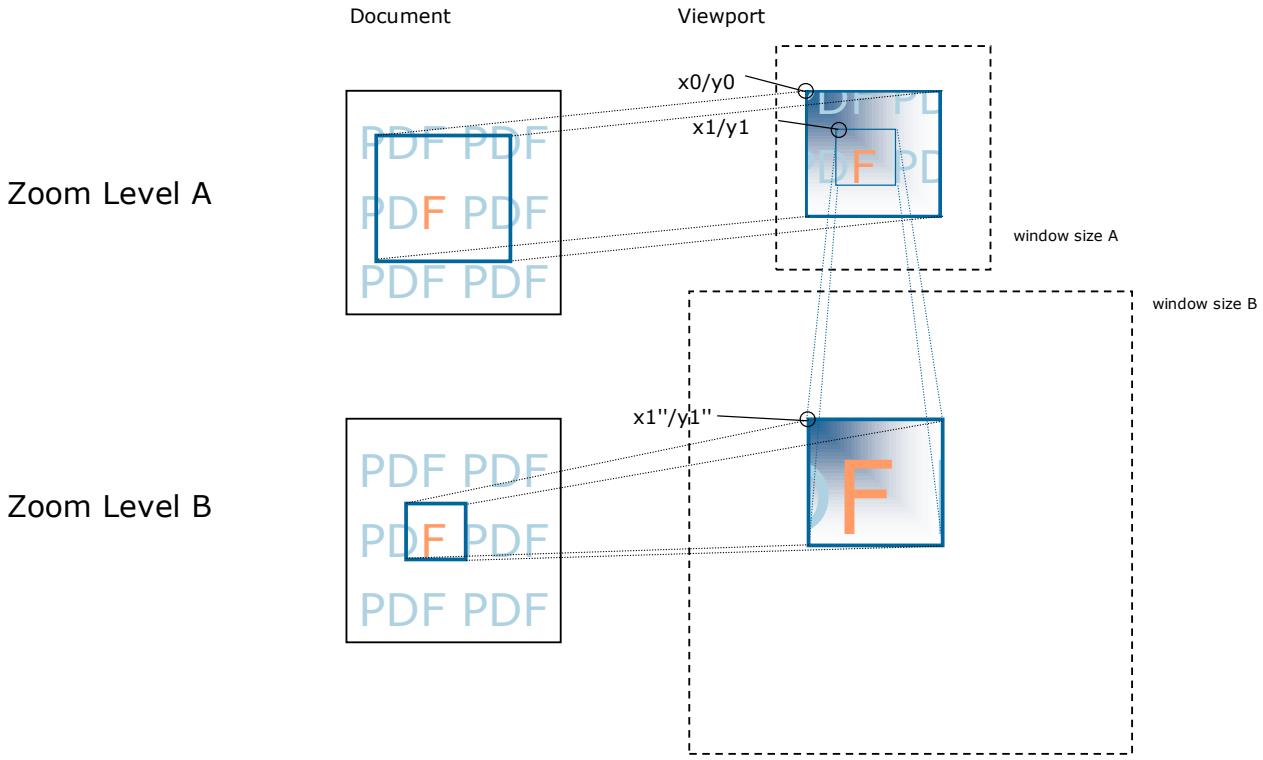
```
class PdfViewerModel extends Model
{
    final static int VIEW_FIX = 1; // Fixed zoom, e.g.\ Actual size
    final static int VIEW_FITWIDTH = 2; // Fit page width into window
    final static int VIEW_FITPAGE = 3; // Fit page width and height

    public void setActualSize()
    {
        if (viewer != null)
            update(HINT_CHANGEVIEW, new Integer(VIEW_FIX));
    }

    public void setFitWidth()
    {
```

```
        if (viewer != null)
            update(HINT_CHANGEVIEW, new Integer(VIEW_FITWIDTH));
    }

    public void setFitPage()
}

class PdfViewerPageView extends JComponent implements View, ActionListener,
    MouseListener
{
    dpiFactor = 96.0f / 72.0f;

    public Dimension getPreferredSize()
    {
        if (page != null)
        {
            final Rectangle rect = page.getWindowRectangle();
2           if (pageView == PdfViewerModel.VIEW_ FITWIDTH)
                zoomFactor = (float)getParent().getWidth() / rect.width
3                   / dpiFactor;
            else if (pageView == PdfViewerModel.VIEW_FITPAGE)
            {
                final float zoomX = (getParent().getWidth()) / rect.width
1                   / dpiFactor;
                final float zoomY = (getParent().getHeight()) / rect.height
                    / dpiFactor;
                zoomFactor = zoomX < zoomY ? zoomX : zoomY;
            }
            final int w = (int) ((float)rect.width * dpiFactor * zoomFactor);
            final int h = (int) ((float)rect.height * dpiFactor * zoomFactor);
            return new Dimension(w, h);
        }
        return getSize();
    }
}
```

5.10 Zoom In and Out

The GUI applications draws the page onto a pane. Zooming in means the window size of that pane size is increased, the viewport size remains, therefore displaying a smaller part of the full page.

1. The model provides three functions related to zooming in and out: Increase current zoom level by 10%, decrease by 10% and set to an absolute value.
2. The model sends the corresponding update to the views (only the `PdfViewerPageView` handles this update).
3. Zooming in and out requires to set a new window and the positioning onto the new window. If one only increases the zoom factor and keeps the coordinates, one does indeed zoom in, however the region displayed is a different one, since the window rectangle has changed. The new position on the new rectangle needs to be calculated. The Code shows one way how to zoom in and out centered.




```
1   class PdfViewerModel extends Model
    {
        final static int HINT_ZOOMIN = 8; // Change the zoom level +10%
        final static int HINT_ZOOMOUT = 16; // Change the zoom level -10%

        public void zoomIn()
        {
            if (viewer != null)
2           update(HINT_ZOOMIN, new Integer(0));
        }

        public void zoomOut()
        {
            if (viewer != null)
2.          update(HINT_ZOOMOUT, new Integer(0));
        }

        public void setZoom(float Zoom)
        {
            if (viewer != null)
            {
                if (Zoom > 0.01f && Zoom < 1000.0f)
2.              update(HINT_SETZOOM, new Float(Zoom));
            }
        }
    }
```

```
class PdfViewerPageView extends JComponent implements View, ActionListener,
    MouseListener
{
    public void update(int hint, Object param)
```

```
    {
        ...
        else if ((hint & PdfViewerModel.HINT_ZOOMIN) != 0 ||
            (hint & PdfViewerModel.HINT_ZOOMOUT) != 0)
        {
            pageView = PdfViewerModel.VIEW_FIX;
            final int zoom = hint;

            // 1. Get (x0, y0) in current window size
            final double x0 = getLocation().getX();
            final double y0 = getLocation().getY();
            final int w = getParent().getWidth(); // width of viewport
            final int h = getParent().getHeight();

3           // 2. Calculate (x1, y1) the new upper left corner after zooming
            final double zoomMod =
                zoom == PdfViewerModel.HINT_ZOOMIN ? 1.1 : 1/1.1;
            final double x1 = x0 - w/2 * (1-1/zoomMod);
            final double y1 = y0 - h/2 * (1-1/zoomMod);
            zoomFactor *= zoomMod;

            // 3. Convert (x1, y1) to ViewPort coordinates (x1', y1')
            AffineTransform trans1 = getTrans(true);
            double[] pt1 = {x1, y1};
            trans1.transform(pt1, 0, pt1, 0, 1);

            // 4. Set size and thereby change the windows size
            setSize(getPreferredSize());

            // 5. Convert (x1', y1') to window coordinates (x1'', y1'')
            AffineTransform trans2 = getTrans(false);
            trans2.transform(pt1, 0, pt1, 0, 1);

            // 6. Set location to (x1'', y1'')
            setLocation((int)pt1[0], (int)pt1[1]);
            repaint();
        }
        else if ((hint & PdfViewerModel.HINT_SETZOOM) != 0)
        {
            pageView = PdfViewerModel.VIEW_FIX;
            zoomFactor = ((Float) param).floatValue();
            setSize(getPreferredSize());
            repaint();
        }
    }
}
```

5.11 Difference View

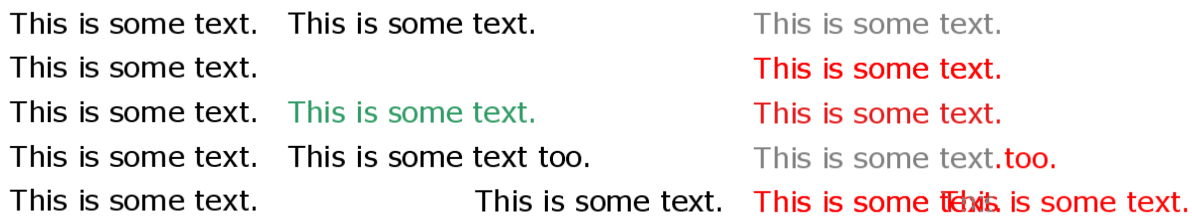
The 3-Heights™ Java Document Viewer 1.8.20.1 and later provides the functionality for a difference view.

The difference view can be used to compare two PDF documents and visualize differences. More concretely it works like this:

- Comparison is done pixel based.
- Pixels that are equal in both documents are converted to grayscale.

- Pixels that are different are displayed in red color, the larger the difference of the two pixels, the higher the intensity of the red color.
- If the two documents have a different amount of pages, the larger amount is taken, pages that are missing in either document are different in every pixel.

Here is an example with two documents:



1. Document **2. Document** **Difference View**

1st Line: No differences, the result is displayed in grayscale
2nd Line: Text is missing, difference is displayed in red
3rd Line: Text color changed, the difference of the color is represented by the intensity of the red
4th Line: The additional word and the moved period are marked red
5th Line: The location of the text is moved, every pixel that is only existing in either document is marked red, if there are any overlapping text portions, they are marked grayscale.

The two new classes to support the difference view are: `PdfDifferencePage` and `PdfDifferenceView`. They are inherited from `PdfPage` and `PdfView`. This allows for using the difference view equally as the view of a normal (PDF) document. See also chapter Open and Close a Document.

Normal Viewer:

```
viewer = new PdfViewer(fileName, password);
```

Difference Viewer:

```
import com.pdftools.pdfviewer.PdfDifferenceViewer;
viewer = new PdfDifferenceViewer(fileName1, pwd1, fileName2, pwd2);
```

5.12 Extract Text

The text extraction interface is used to access the text contained in a page. Each individual text word and its properties such as the text string, position, bounding rectangle and the font will be returned in the order in which they appear in the page's content stream (so called z-order).

```
public void extractTextFromPage(PdfViewerPage page)
{
    PdfTextExtractor extractor = page.getTextExtractor();
    for (;;)
    {
        PdfText text = extractor.getNextText();
        if (text == null)
            break;
        String s = text.getString();
        ...
    }
}
```

In order to get access to the text of the whole document the pages can be enumerated first.

```
public void extractTextFromDocument(PdfViewer viewer)
{
    for (int i = 1; i < viewer.getPageCount(); i++)
    {
        PdfViewerPage page = viewer.getPage(i);
        extractTextFromPage(page);
    }
}
```

The position property of the text can be used extract text in a specific range e.g. a mark rectangle. The coordinates are returned in point units of the document's coordinate system. To get the viewport coordinates use the page's window to viewport transform method.

```
Point2D pt = text.getPosition();
if (is_in_range(pt))
...
```

5.13 Find Text

All occurrences of a text which matches a given regular expression, e.g. all words ending with "is", can be found using the text finder interface.

```
public void findTextInPage(PdfViewerPage page)
{
    Color clr = new Color(255, 255, 255);
    PdfTextFinder finder = page.getTextFinder(".*?is", clr);
    for (;;)
    {
        PdfText text = finder.getNextText();
        if (text == null)
            break;
        ...
    }
}
```

Finding text in the whole document is made easier using the page finder interface instead of just enumerating the pages and then using the text finder. This has the advantage that only those pages with at least one occurrence are returned.

```
public void findTextInDocument(PdfViewer viewer)
{
    Color clr = new Color(255, 255, 255);
    PdfPageFinder pages = viewer.getPageFinder(".*?is", clr);
    for (;;)
    {
        PdfTextFinder finder = pages.getNextFinder();
        for (;;)
        {
            PdfText text = finder.getNextText();
            if (text == null)
               break;
```

```
            ...
        }
    }
}
```

All occurrences of the found text can be high-lighted on the page display using the draw method of the text finder.

```
public void draw(Graphics2D g, Rectangle rect, PdfViewerPage page,
    PdfTextFinder finder)
{
    page.draw(g, rect);
    finder.draw(g, rect);
}
```

If one needs to navigate to a specific occurrence of a text word then the position property and some context information can be used to accomplish this.

```
PdfTextFinder finder = page.getTextFinder();
...
PdfText text = finder.getNextText();
...
Point2D pos = text.getPosition();
goTo(finder.getPageNo(), pos.getX(), pos.getY(), zoom);
...
```

6 Licensing, Copyright, and Contact

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

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

Licensing and Copyright

Contact

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

A Appendix

A.1 Font Replacement Strategy

1. A Type3 font is expanded inline. Glyphs are cached.
2. If the font is embedded and the `eOptionOutline` is turned on then the glyphs are converted to graphics paths.
3. If the font is embedded and is installable it will be converted to the windows format and temporarily installed. While printing the font is converted to a TrueType font prior to being installed.
4. If the font is not embedded and it is a standard font then a standard font in the fonts directory (ZapfDingbats) or a pre-installed font is used instead.
5. If the font is not a standard font and it is a pre-installed font then the pre-installed font is used.
6. If the font is not a pre-installed font and it is listed in the `fonts.ini` file then the replacement font is used.
7. If the font is not listed in the fonts.ini file the nearest pre-installed font is used depending on the font's properties (family, pitch, Unicode subset, weight, italic, etc.).