## Contents

1 **Introduction** .................................................................................................................. 4  
1.1 Description ....................................................................................................................... 4  
1.2 Functions ......................................................................................................................... 4  
1.2.1 Features ..................................................................................................................... 4  
1.2.2 Formats ...................................................................................................................... 4  
1.2.3 Conformance ............................................................................................................... 5  
1.3 Operating Systems .......................................................................................................... 5  

2 **Installation** ...................................................................................................................... 6  
2.1 Preparation for OCR Processing ..................................................................................... 6  
2.1.1 ABBYY FineReader Engine ...................................................................................... 6  
2.1.2 3-Heights™ OCR Service ....................................................................................... 6  
2.2 Installation of the 3-Heights™ Scan to PDF Server ....................................................... 6  
2.2.1 Installed Files ........................................................................................................... 6  
2.3 Required Licenses .......................................................................................................... 7  
2.4 Special Directories .......................................................................................................... 7  
2.4.1 Directory for temporary files .................................................................................... 7  
2.4.2 Cache Directory ....................................................................................................... 7  
2.4.3 Font Directories ........................................................................................................ 8  

3 **License Management** ...................................................................................................... 9  
3.1 License Installation and Management ........................................................................... 9  
3.1.1 Graphical License Manager Tool ............................................................................ 9  
3.1.2 Command Line License Manager Tool .................................................................. 10  
3.2 License Selection and Precedence .................................................................................. 11  
3.2.1 Selection .................................................................................................................. 11  
3.2.2 Precedence ............................................................................................................... 11  
3.3 Key Update ..................................................................................................................... 12  
3.4 License activation .......................................................................................................... 12  
3.4.1 Activation ................................................................................................................ 12  
3.4.2 Reactivation .............................................................................................................. 13  
3.4.3 Deactivation ............................................................................................................ 13  
3.5 Proxy Setting .................................................................................................................. 13  
3.6 Offline Usage .................................................................................................................. 14  
3.6.1 First Step: Create a Request File ............................................................................ 14  
3.6.2 Second Step: Use Form on Website ....................................................................... 15  
3.6.3 Third Step: Apply the Response File ..................................................................... 15  
3.7 License Key Versions .................................................................................................... 15  
3.8 License Key Storage ....................................................................................................... 15  
3.8.1 Windows .................................................................................................................. 15  
3.9 Troubleshooting ............................................................................................................. 16  
3.9.1 License key cannot be installed .............................................................................. 16  
3.9.2 License is not visible in license manager .............................................................. 16  
3.9.3 License is not found at runtime ............................................................................ 16  
3.9.4 Eval watermark is displayed where it should not .............................................. 17  
3.9.5 Activation is not recognized .................................................................................. 17  
3.9.6 Activation is invalidated too often ......................................................................... 18  
3.9.7 Connection to the licensing service fails .............................................................. 18  
3.9.8 Offline usage fails due to a request/response mismatch ....................................... 18
1 Introduction

1.1 Description

The 3-Heights™ Scan to PDF Server is a service based application for scan post processing that is modular, flexible and highly parallel.

1.2 Functions

1.2.1 Features

- Conversion of single page or multi-page raster images to PDF
- Processing of subfolders
- Flexible workflow configuration
- Set output format and conformity level (PDF, PDF/A-1, PDF/A-2 and PDF/A-3)
- Optical character recognition (OCR) including barcodes
- Digital PDF signature
- Parallel processing

Compression

- Set image compression individually different classes of images
- Support for mixed raster content (MRC)
- CCITT Group 3 (1D and 2D)
- CCITT Group 4
- LZW
- JPEG
- Deflate (ZIP)
- JPEG2000
- JBIG2 (lossless only)

1.2.2 Formats

Input Formats

- JPEG
- TIFF
- scanned PDF

Output Formats

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

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

1.3 Operating Systems

The 3-Heights™ Scan to PDF Server is available for the following operating systems:
- Windows Client 7+ | x86 and x64

‘+’ indicates the minimum supported version.
2 Installation

2.1 Preparation for OCR Processing

The following products are necessary to use OCR processing:

2.1.1 ABBYY FineReader Engine

The ABBYY FineReader Engine provides the actual OCR recognition.

Detailed installation instructions can be found in the following manuals:

- 3-Heights™ OCR Add-On for ABBYY FineReader Engine v11: [OcrAbbyy11.pdf](#)
- 3-Heights™ OCR Add-On for ABBYY FineReader Engine v12: [OcrAbbyy12.pdf](#)

The most important steps are:

1. Installation of the actual ABBYY FineReader Engine 11 or 12.
2. Activation of a valid license key for the ABBYY FineReader Engine in the ABBYY License-Manager.

2.1.2 3-Heights™ OCR Service

The 3-Heights™ OCR Service acts as a proxy between the 3-Heights™ Scan to PDF Server and the ABBYY FineReader Engine.

Detailed installation instructions can be found in the following manual: [OcrService.pdf](#)

The most important steps are:

1. Installation of the 3-Heights™ OCR Service.
2. Installation of a valid license for the 3-Heights™ OCR Service.
3. Configuration of the 3-Heights™ OCR Service in the OcrSvr.xml configuration file. Especially the default-plugin attribute must match the version of the ABBYY FineReader Engine.

2.2 Installation of the 3-Heights™ Scan to PDF Server

To install the 3-Heights™ Scan to PDF Server, simply run the MSI installer `Scan2PdfServer-<version>-Windows.msi`.

2.2.1 Installed Files

The following files are installed in `C:\Program Files (x86)\PDF Tools AG\3-Heights(TM) Scan to PDF Server`:

### General Files (required)

- `\ScanServer.exe`
- `\Scanserver.exe.config` (persistent)
- `\ScanServer.xml` (persistent)
- `\LicenseManager.exe`
2.3 Required Licenses

The following licenses are necessary for the 3-Heights™ Scan to PDF Server:

- 3-Heights™ Scan to PDF Server
- 3-Heights™ Image to PDF Converter Shell (for TIFF processing)
- 3-Heights™ PDF Security Shell (for PDF signing)
- ABBYY FineReader Engine (for OCR processing only)
- 3-Heights™ OCR Service (for OCR processing only)

2.4 Special Directories

2.4.1 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.

2.4.2 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**.

### 2.4.3 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. User fonts listed in the registry key \HKEY_CURRENT_USER\Software\Microsoft\Windows NT\CurrentVersion\Fonts. This includes **user specific fonts from C:\Users\<user>\AppData\Local\Microsoft\Windows\Fonts** and **app specific fonts from C:\Program Files\WindowsApps**
3. **directory Fonts**, which must be a direct sub-directory of where SCSE.exe resides.
3 License Management

The 3-Heights™ Scan to PDF Server requires a valid license in order to run correctly.

3.1 License Installation and Management

There are two 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.

3.1.1 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` shell tool is available in the Utilities & Tools section of your My PDF Tools customer account.

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 an * 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-XXXXX
```

Delete old license key:

```
licmgr delete 1-XXXXX-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-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 selected for the current user
2. License selected for the current user (legacy key format)
3. License selected for all users
4. 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 updated directly in the license manager.

**In the Graphical User Interface** select the license and press the button “Update Key” in the toolbar:

![License Manager Interface]

**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:

![License Manager Interface]

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-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)
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.
- The software was not restarted after registering the license.

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.
- 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.

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

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.
- 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:
- One of the properties used to calculate the system fingerprint is changing frequently.

Solution
Update to a newer version of the PDF Tools product, deactivate the license key using the new license manager and activate it again. After that, an improved 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.

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 Configuration

4.1 Application configuration

The application configuration can be changed in the file ScanServer.exe.config inside the installation directory, but the default configuration is usually sufficient.

The application configuration is very simple and provides only the following properties:

4.1.1 Configuration

Property (get, set): String Configuration
Default: "ScanServer.xml"

Path to the process definition file. The path is evaluated relative to the application configuration file.

4.1.2 ConfigurationLog

Property (get, set): String ConfigurationLog
Default: "configuration.log"

Path to the configuration log file. The path is evaluated relative to the application configuration file.

This file is written if an error occurs while starting the service. After the service startup, logging is defined by the Process definition file.

4.2 Process definition

The process definition file is usually called ScanServer.xml and located in the installation directory. However, the name and location can be overridden in the Application configuration file.

The process definition consists of so-called components. Every component provides a certain functionality and also defines how it is configured.

All components are explained in detail in the Component Reference. The main/root component has always the type and name ScanServer.Root.

A list of complete examples is provided in the Cook book.
5 Component Reference

5.1 Main components

5.1.1 ScanServer.Root

This is the main component in the Scan to PDF Server product.

XML configuration structure

<Workfolder> The folder for temporary files.
  path The path to the workfolder.

<Logging> A component of type Log.Container defining the logging facility. This setting can be overridden by specific Groups.

<Group> (1..n) A group for processing similar files.
  <InputFolder> The watched folder for the input files.
    path The path to the folder.
    filter (optional) A pattern for filtering filenames.
    folders (optional) If "true", The input folder is scanned for folders instead of files.
      Default value: "false".
    trigger (optional) If folders="true", this denotes a pattern of files inside a folder, that will trigger the processing of the containing folder.
    lock-files (optional) If "true", the files in the input directory are locked during the entire process to prevent them from being modified or removed.
      Default value: "false".

Note: This option should not be set to "true" for bulk processing of many input files at once, as the process might run out of file handles.

<OutputFolder> The output folder for the resulting files in case of success.
  path The path to the folder.
  suffix (optional) Suffix used for name conflict resolution. The placeholder {0} is replaced by an automatic index number.
    Default value is " {{0}}"

append-suffix
  "never" Suffix is never appended. Name conflicts result in an error.
  "always" Suffix is always appended.
  "auto" (default) Suffix is only appended in case of naming conflicts

<FailedFolder> The output folder for the original files in case of an error.
  path The path to the folder.
<Logging> (optional) A component of type Log.Container overriding the global setting.

<Processing> The processing pipeline, a component of type Composition.Linear. Input item type must be IFile. Output item type can be either IFile or IList<IFile>.

Example:

```xml
<ScanServer.Root>
  <Workfolder path="C:\Scan Server\Workfolder"/>
  <Logging/>
  <Group>
    <InputFolder path="C:\Scan Server\Input" filter="*.tif"/>
    <OutputFolder path="C:\Scan Server\Output"
      suffix="{}.0" append-suffix="auto"/>
    <FailedFolder path="C:\Scan Server\Failed"/>
    <Logging/>
    <Processing/>
  </Group>
  <!-- Logging configuration is omitted here -->
  <Logging/>
  <Group/>
  <!-- Another group -->
</ScanServer.Root>
```

5.2 Composition of components

Components from the namespace Composition provide functionality for combining other Components.

5.2.1 Loading subcomponents

If some component can have subcomponents of arbitrary type, there must be some a mechanism to specify this type.

There are two variants to specify the type of a component:

**Short hand form**

For components defined in the Scan Server itself, the type can be specified in the element name like:

```
<Namespace.Type>
```

This is a short hand for general form:

```
<... c:type="ScanSolution.Components.Namespace.Type"/>
```
**General Form**

Components can be loaded from a different .NET assembly by using the .NET fully qualified name:

```xml
<... xmlns:c="http://www.pdf-tools.com/scanserver/composition/container"
c:type="Namespace1.Namespace2.Type, AssemblyName">
```

**Namespace form**

If many components from the same assembly are used, the namespace form comes handy.

On the root XML element (`<ScanServer.Root>`) a mapping from XML namespaces to .NET namespaces can be defined:

```xml
<ScanServer.Root
 xmlns:c="http://www.pdf-tools.com/scanserver/composition/container"
 xmlns:p1="http://www.example.com/plugin"
 xmlns:n="http://www.example.com/plugin Example.Namespace, ExampleAssembly">
```

This can later be referenced like:

```xml
<p1:PluginComponent>
```

**Example:**

This example combines two components operating on a file into a new custom component, also operating on a file:

```xml
<!-- Components from the Scan Server application with short form
     __________________________________________________________ -->
<Tiff.Ocr>
    <!-- OCR configuration is omitted here -->
</Tiff.Ocr>
<Tiff.Compress>
    <!-- Compression configuration is omitted here -->
</Tiff.Compress>
<Log.File>
    <!-- Log file configuration is omitted here -->
</Log.File>

<!-- Same components in general form (Element name is not important)
     __________________________________________________________ -->
<TiffOcr c:type="ScanSolution.Components.Tiff.Ocr"/>
    <!-- OCR configuration is omitted here -->
</TiffOcr>
<TiffCompress c:type="ScanSolution.Components.Tiff.Compress">
    <!-- Compression configuration is omitted here -->
</TiffCompress>
    <!-- Log file configuration is omitted here -->
</LogFile>

<!-- Plugin component
     __________________________________________________________ -->
<Custom c:type="CustomPlugin.CustomNamespace.CustomComponent, CustomPlugin">
```
5.2.2 Composition.Linear

This component provides functionality for chaining together multiple components.

It takes a list of components and connects the output of each component with the input of the next component in the chain. The input of the composed component is the input of the first component, the output is taken from the last component. The type of the component is determined according to chapter Loading subcomponents.

The item types of the subcomponents will be inferred. If the types do not match, the configuration is rejected.

**Input item type**

The input item type is the same as the input item type of the first subcomponent.

**Output item type**

The output item type is the same as the output item type of the last subcomponent.

**XML configuration structure**

<...> (1..n) Subcomponent, see Loading subcomponents.

**Example:** This example combines two components operating on a file into a new custom component, also operating on a file.

```xml
<Composition.Linear>
  <Tiff.Ocr>
    <!-- OCR configuration is omitted here -->
  </Tiff.Ocr>
  <Tiff.Compress>
    <!-- Compression configuration is omitted here -->
  </Tiff.Compress>
</Composition.Linear>
```

5.3 List processing

The components from the List namespace provide functionality for parallel processing of a list of items.

5.3.1 List.Map

With the List.Map component, an operation can be applied to a list of items in parallel. The result is again a list of items.

The content of that component is itself a component of type Composition.Linear.
XML configuration structure

See Composition.Linear.

Example: This example is the same as in Composition.Linear, but instead of operating on a single file, it operates on a list of files.

```
<List.Map>
  <Tiff.Ocr>
    <!-- OCR configuration is omitted here -->
  </Tiff.Ocr>
  <Tiff.Compress>
    <!-- Compression configuration is omitted here -->
  </Tiff.Compress>
</List.Map>
```

5.3.2 List.Filter

This component provides functionality for filtering a list of items. The content is again a component of type Composition.Linear, with the restriction that the output type must be a IBool.

Input item type

IList<TItem> where TItem can be any type of item.

Output item type

IList<TItem> (Same type as input item)

XML configuration structure

See Composition.Linear.

Example: Example for filtering out empty files with a custom plugin.

```
<List.Filter>
  <Empty c:type="CustomPlugin.IsEmptyFile, CustomPlugin">
    <!-- Configuration of custom component is omitted here -->
  </Empty>
  <Bool.Not/>
</List.Filter>
```

5.3.3 List.SortBy

Sort the list by a dynamically computed key.

The content is again a component of type Composition.Linear, with the restriction that the output type must be a IString.
Input item type

$IList<TItem>$ where $TItem$ can be any type of item.

Output item type

$IList<TItem>$ (Same type as input item)

XML configuration structure

See $Composition.Linear$. Additional Attributes:

comparison The string comparison type:

"current-culture" (Default) Compare strings using culture-sensitive sort rules and the current culture.
"current-culture ignore-case" Compare strings using culture-sensitive sort rules, the current culture, and ignoring the case of the strings being compared.
"invariant-culture" Compare strings using culture-sensitive sort rules and the invariant culture.
"invariant-culture ignore-case" Compare strings using culture-sensitive sort rules, the invariant culture, and ignoring the case of the strings being compared.
"ordinal" Compare strings using ordinal (binary) sort rules.
"ordinal ignore-case" Compare strings using ordinal (binary) sort rules and ignoring the case of the strings being compared.
"logical" Compare strings using logical sort rules and ignoring the case of the strings being compared. Digits in the strings are considered as numerical content rather than text. The result can differ between different version of Microsoft Windows. It should not be used for canonical sorting applications. This is the sort rule used by the Windows Explorer.

Example: Sort the files in a list in the same order as Windows Explorer.

```xml
<List.SortBy comparison="logical">
  <File.Name/>
</List.SortBy>
```

5.3.4 List.ChunkIf

This component provides functionality for splitting a list of items into chunks. The content is again a component of type $Composition.Linear$, with the restriction that the output type must be a $IBool$.

If the inner pipeline for an element returns $true$, then the list is split before that element.

Input item type

$IList<TItem>$ where $TItem$ can be any type of item.

Output item type

$IList<IList<TItem>>$ where $TItem$ is the same type as $TItem$ of the input item type.
XML configuration structure

See Composition.Linear.

Example: Example for splitting by barcode using a custom plugin.

```
<List.ChunkIf>
  <Barcode c:type="CustomPlugin.ContainsBarcode, CustomPlugin">
    <!-- Configuration of custom component is omitted here -->
  </Barcode>
</List.ChunkIf>
```

5.3.5 List.Join

Join a nested list into an unnested list.

This component performs the inverse functionality of List.ChunkIf.

Input item type

```
IList<IList<TItem>> where TItem can be any type of item.
```

Output item type

```
IList<TItem> where TItem is the same type as TItem of the input item type.
```

Example: Split a list of TIFFs into pages and concatenate all pages into a single list.

```
<Folder.Files filter="*.tif"/>
<List.Map>
  <Tiff.SplitPages/>
</List.Map>
<List.Join/>
```

5.3.6 List.IsEmpty

Check if a list is empty.

Input item type

```
IList<IItem> (A list of items of any kind)
```

Output item type

```
IBool
```
XML configuration structure

This component has no properties that can be configured.

Example:

```xml
<List.IsEmpty/>
```

5.3.7 List.ElementAt

Extract a single element from a list.

Input item type

```csharp
IList<TItem> (A list of items of a specific type)
```

Output item type

```csharp
TItem (A single item of the same specific type)
```

XML configuration structure

```csharp
index The index of the element to be extracted (starting with 0).
```

Example:

```xml
<List.ElementAt index="0"/>
```

5.4 TIFF processing

The components from the Tiff namespace provide functionality for processing TIFF files.

5.4.1 Tiff.ImportImage

The Tiff.ImportImage imports the image data from other file formats into a TIFF file, if possible without changing or recompressing the image data.

Supported file formats

- JPEG File Interchange Format (*.jpg, *.jpeg, *.jpe)
- PDF (*.pdf): Supported are only PDFs as they are typically produced by scan stations, i.e. only PDFs that contain a single image and nothing else.

Input item type

The input item type of this component is IFile.
Output item type

The output item type of this component is IFile.

XML configuration structure

default-resolution  The image resolution in DPI, if not already defined in the image.

Example:

```xml
<Tiff.ImportImage default-resolution="96"/>
```

5.4.2 Tiff.SplitPages

The Tiff.SplitPages component splits a TIFF file into a list of TIFF files, each containing only one page of the input file.

Input item type

The input item type of this component is IFile.

Output item type

The output item type of this component is IList<IFile>.

XML configuration structure

This component has no properties that can be configured.

Example:

```xml
<Tiff.SplitPages/>
```

5.4.3 Tiff.Merge

The Tiff.Merge component merges a list of TIFF files into a single TIFF file.

Input item type

The input item type of this component is IList<IFile>.

Output item type

The output item type of this component is IFile.
**XML configuration structure**

**remove-blank-pages**  Set `remove-blank-pages="true"` to remove blank pages previously recognized by the Tiff.Ocr component. Default value is `remove-blank-pages="false"`.

**Example:**

```xml
<Tiff.Merge/>
```

### 5.4.4 Tiff.Ocr

This component analyzes the TIFF and stores additional information in the file. This information can later be used by other components:

- Recognized text can be used by the Tiff.ConvertToPdf component to generate a text layer in the PDF.
- Recognized barcodes can be used to split the document. See example Splitting a TIFF by barcode for more details.
- Recognized picture regions can be used by the Tiff.Compress component to improve the MRC compression result.
- Recognized mask image can be used by the Tiff.Compress component to improve the MRC compression result.

**Input item type**

The input item type of this component is **IFile**.

**Output item type**

The output item type of this component is **IFile**.

**XML configuration structure**

- **plugin**  The OCR plugin used for recognition. Possible values: See Available OCR plugins
- **parameters**  The parameters that are passed to the plugin. The format of the value is plugin-specific.
- **languages**  The languages that are passed to the plugin. The format of the value is plugin-specific.
- **binarize**  Set `binarize="true"` if the image should be converted to black/white before processing. Default value is `binarize="false"`.
- **max-parallel**  Limit the number of concurrent OCR tasks. Default value is `max-parallel="-1"` (unlimited).

**Available OCR plugins**

OCR engines are accessed through the corresponding OCR plugins. At present the following OCR plugins are supported:

"**internal**"  Internal Image Analysis Engine
The internal engine can only recognize the following element:

- Picture regions.
  Use OCR parameter RecognizePictures=true to enable.
- Mask image.
  Use OCR parameter DisableMaskEmbedding=true to disable.
- Blank pages.
  Use OCR parameter RecognizeBlankPages=true to enable.

**Note:** Text recognition is **not** supported by this engine.

"service"  3-Heights™ OCR Service

Access one of the supported OCR engines over a proxy service.

The service point can be specified directly as "service@http://hostname:7982/" or by editing the configuration file ocrserver.ini.  
See the manual of the 3-Heights™ OCR Service for more details.

"abbyy11"  Abbyy FineReader 11 OCR Engine

"abbyy10"  Abbyy FineReader 10 OCR Engine

Example:

```xml
<Tiff.Ocr plugin="service"
  languages="German,English"
  parameters="PredefinedProfile=DocumentConversion_Accuracy"/>
```

### 5.4.5 Tiff.ExtractOcrData

Extract the OCR data that was previously embedded by Tiff.Ocr.

The extracted information is a list of XML files, one for each page of the TIFF. The XML format is defined in Appendix A.  
To be forward compatible with future versions, it is strongly recommended to use the **format-version** attribute.

**Input item type**

The input item type of this component is a TIFF file (IFile).

**Output item type**

The output item type of this component is a list of XML files (IList<IFile>), one for each page of the TIFF.

**XML configuration structure**

- **format-version**  The version of the XML format.
  Possible values: "1", "2" or "3". Default: As stored in the TIFF (no reformatting).

Example:

```xml
<Tiff.ExtractOcrData/>
```
5.4.6 **Tiff.Compress**

Provides functionality for compressing a TIFF file and perform MRC segmentation.

**XML configuration structure**

- **recompress-lossy**  Recompress JPEG, JPEG2000 and JBIG2 streams.
  Default value: recompress-lossy="true"
- **upgrade-jpeg**  Upgrade from JPEG 6 to JPEG (Technote 2).

**<Bilevel-Compression>**  Compression settings used for bilevel (black/white) images.
  - **type**  Compression type (Possible values see below)
  - **quality**  Compression quality for lossy compression (1..100)

**Grayscale-Compression**  Compression settings used for grayscale images.
  - **type**  Compression type (Possible values see below)
  - **quality**  Compression quality for lossy compression (1..100)

**<Lab-Compression>**  Compression settings used for images in the CIE L*a*b* color space.
  - **type**  Compression type (Possible values see below)
  - **quality**  Compression quality for lossy compression (1..100)

**<Paletted-Compression>**  Compression settings used for images with a color palette.
  - **type**  Compression type (Possible values see below)
  - **quality**  Compression quality for lossy compression (1..100)

**<RGB-Compression>**  Compression settings used for RGB (and YCbCr) images.
  - **type**  Compression type (Possible values see below)
  - **quality**  Compression quality for lossy compression (1..100)

**<Separated-Compression>**  Compression settings used for images in a separated color space (including CMYK).
  - **type**  Compression type (Possible values see below)
  - **quality**  Compression quality for lossy compression (1..100)

**<MRC>**  Mixed raster content settings
  - **segmentation**  If segmentation="true", mixed raster content segmentation is performed.
  - **mask-threshold**  Binarization threshold for MRC segmentation.
    Default: automatic

**<MaskFill-Compression>**  Compression of the fill image of the background and foreground layer.
  - **type**  Compression type (Possible values see below)
  - **quality**  Compression quality for lossy compression (1..100)
  - **scale**  Reduce the resolution by a factor.

**Compression types**

The following values are possible for the `type` attribute on `<XXX-Compression>` elements:
"raw" Uncompressed
"group3" CCITT Group 3 Fax Compression
"group3-2d" CCITT Group 3 (2D) Fax Compression
"group4" CCITT Group 4 Fax Compression
"lzw" LZW Compression (GIF)
"jpeg" JPEG (Technote 2) Compression
"jpeg6" JPEG 6 Compression
"deflate-adobe" Adobe Deflate Compression
"packbits" Packbits Compression
"deflate" Deflate Compression (ZIP)
"jpeg2000" JPEG 2000 Compression
"jbig2" JBIG 2 Compression (lossless only)

Example:

```xml
<Tiff.Compress upgrade-jpeg="true"
  recompress-lossy="false">
  <Bilevel-Compression type="group4"/>
  <Grayscale-Compression type="jpeg" quality="90"/>
  <RGB-Compression type="jpeg" quality="75"/>
  <MRC segmentation="true">
    <MaskFill-Compression type="jpeg" quality="20" scale="2"/>
  </MRC>
</Tiff.Compress>
```

### 5.4.7 Tiff.ConvertToPdf

Provides functionality for converting a list of TIFF files into a PDF document.

**XML configuration structure**

**compliance** The conformance level of the generated PDF.
Supported values are:

- "pdf1.x" Regular PDF Versions such as 1.4, 1.5, 1.6, 1.7
- "pdf2.0" Regular PDF Version 2.0
- "pdfa-1b" PDF/A-1b format
- "pdfa-1a" PDF/A-1a format (accessibility)
- "pdfa-2b" PDF/A-2b format
- "pdfa-2u" PDF/A-2u format (unicode)
- "pdfa-2a" PDF/A-2a format (accessibility)
- "pdfa-3b" PDF/A-3b format
- "pdfa-3u" PDF/A-3u format (unicode)
- "pdfa-3a" PDF/A-3a format (accessibility)
**default-resolution**  The image resolution in DPI, if not defined in the image. (Default: 96)

**user-unit**  The user unit in multiples of points.

**mrc-layers**  Create optional content groups (PDF layers) for MRC layers.
  Default: "false"

**ocr-text**  Create PDF text from embedded OCR information.
  Default: "true"

**output-intent-path**  Path to the desired output intent profile.
  Example: "C:\Windows\System32\spool\drivers\color\sRGB Color Space Profile.icm"

**fallback-color-profile-path**  Path to the desired fallback color profile. This profile is used for TIFFs without embedded color profile.
  Example: "C:\Windows\System32\spool\drivers\color\sRGB Color Space Profile.icm"

**xmp-path**  Path to the desired XMP metadata file.

**Example:**

```
<Tiff.ConvertToPdf mrc-layers="true"/>
```

## 5.5 PDF processing

### 5.5.1 Pdf.Sign

Sign a PDF document.

A detailed description of all options is provided by the manual of the 3-Heights™ PDF Security Shell Tool.

**Input item type**

The input item type of this component is **IFile**.

**Output item type**

The output item type of this component is **IFile**.

**XML configuration structure**

- **provider**  The cryptographic provider used to sign the document.
- **store**  The name of the certificate store (Microsoft CryptoAPI provider only)
- **location**  The location of the certificate store (Microsoft CryptoAPI provider only)

```
<SessionProperty>  (1..n) Generic, provider specific session properties
  name  The name of the property
  type  The type of the property. Possible types are "string" and "file"
  value  The value of the property
```

- **TimeStamp**  The time stamp server configuration
url  The url of the time stamp server

credentials  The credentials for accessing the time stamp server.

<WebProxy>  The web proxy server configuration
  url  The URL of the web proxy server
  credentials  The credentials for accessing the web proxy server.

<Certificate>  Certificate selection
  name  The (proper) name of the certificate ("Issued to")
  issuer  The issuer of the certificate ("Issued by")
  serial  The serial number of the certificate
  fingerprint  The certificate fingerprint

<Signature>  
  reason  The signature reason
  contact-info  Contact info of the signer

<Appearance>  
  page  The page number of the signature appearance
  bounding-box  The bounding box of the signature appearance

<Border>  The border appearance
  color  The color of the border (color name or #RRGGBB)
  width  The line width of the border

<Background>  The border appearance
  color  The color of the background (color name or #RRGGBB)
  image  Path to the background image

<Text1>  The first text line The inner text of this element is used as first line
  font  The name of the font or path to the font file.

<Text1>  The second text line The inner text of this element is used as second line
  font  The name of the font or path to the font file.

Example:

```xml
<Pdf.Sign>
  <TimeStamp url="http://tsa.swisssign.net"/>
  <Certificate name="Hans Muster"
    issuer="Example CA"/>
  <Signature reason="I'm the author of this document"
    contact-info="Please do not contact me">
    <Appearance>
      <Border color="blue" width="10"/>
      <Background color="#FF0000"/>
      <Text1 font="CourierNew">I signed this document</Text1>
      <Text1 font="Helvetica">I'm the author</Text1>
    </Appearance>
</Pdf.Sign>
```
5.6 XML processing

5.6.1 Xml.Parse

Parse an XML file for further processing.

**Input item type**

The input item type of this component is **IFile**.

**Output item type**

The output item type of this component is an XML node (**XmlNode**).

**XML configuration structure**

This component has no properties that can be configured.

**Example:**

```
<Xml.Parse/>
```

5.6.2 XPath.Select

Select XML nodes with an XPath expression.

**Input item type**

The input item type of this component is **XmlNode**.

**Output item type**

The output item type of this component is a list of XML nodes (**IList<XmlNode>**).

**XML configuration structure**

This component contains the XPath-Expression as text. Namespaces can be declared on the element itself.

**Example:** Select all barcode nodes in an OCR-XML

```
<XPath.Select xmlns:ocr="http://www.pdf-tools.com/ocr">
    //ocr:page/ocr:page-content/ocr:barcode
</XPath.Select>
```
5.7 Logging

Components from the namespace Log provide logging functionality.

5.7.1 Reporting level

All logging components have a property called reporting-level, that specifies which type of log entries are actually logged.

Possible values are:
- "error" Errors only.
- "warning" Warnings and errors.
- "info" All types.

5.7.2 Log.Container

The Log.Container component bundles different logging facilities. Log entries are routed to all subcomponents in parallel.

XML configuration structure

reporting-level See Reporting level.

details If "true", additional information is logged for each log entry (if available). The default value is "false".

<i>...</i> (1..n) Logging subcomponents, see Loading subcomponents.

Example:

```xml
<Log.Container reporting-level="warning">
  <LogFile>
    <!-- Log file configuration is omitted here -->
  </LogFile>
  <EventLog>
    <!-- Windows event log configuration is omitted here -->
  </EventLog>
</Log.Container>
```

5.7.3LogFile

The LogFile component provides the functionality to log into a file.

XML configuration structure

reporting-level See Reporting level.
**details** If "true", additional information is logged for each log entry (if available). The default value is "false".

**path** The path to the logfile. If the path contains the character sequence `{0}`, this sequence is replaced with an integer number on every startup, to make the logfile unique.

Example:

```xml
<LogFile reporting-level="warning" path="C:\Scan Server\log-{0}.txt"/>
```

### 5.7.4 EventLog

The **EventLog** component provides the functionality to log to the Windows event log.

**XML configuration structure**

- **reporting-level** See **Reporting level**.
- **source** The event log source.

Example:

```xml
<EventLog reporting-level="warning" source="ScanServer"/>
```

### 5.8 File System

Components that deal with folders, files and file names.

#### 5.8.1 File.Name

Get the filename of a file.

**Input item type**

The input item type of this component is **IFile**.

**Output item type**

The output item type of this component is **IString**.

**XML configuration structure**

- **extension** Specify, whether the file extension is included. The default value is "true".

Example:

```xml
<File.Name extension="false"/>
```
5.8.2 Folder.Files

Get the files contained in a folder

**Input item type**
The input item type of this component is `IFolder`.

**Output item type**
The output item type of this component is `IList<IFile>`.

**XML configuration structure**

- **filter**  Filter the by filename using wildcards (*, ?). The filter expression is case-insensitive. Multiple filters can be specified separated by a semicolon (;).

- **regex**  Filter the by filename using a regular expression (.NET style). The filter expression is case-insensitive.

**Note:** If both, `filter` and `regex` are specified, `regex` takes precedence.

**Example:**

```xml
<Folder.Files filter="*.tif;*.jpg"/>
```

5.8.3 FileList.AppendSuffix

Rename a list of files, allowing to use the index in the list.

**XML configuration structure**

The configuration contains the suffix as text. The string `{0}` is replaced by the index of the file in the list.

**Example:**

```xml
<FileList.AppendSuffix>{0}</FileList.AppendSuffix>
```
5.9 Boolean Logic

Components that deal with boolean logic.

5.9.1 Bool.Not

Invert a boolean value.

XML configuration structure

This component cannot be configured further.

Example:

```xml
<Bool.Not/>
```

5.10 Complete example

```xml
<ScanServer>
  <WorkFolder path="C:\Scan Server\Workfolder"/>
  <Logging>
    <LogFile path="C:\Scan Server\log-{0}.txt" reporting-level="warning"/>
    <EventLog source="ScanServer" reporting-level="info"/>
  </Logging>
  <!-- Compress TIFF input
  =============== -->
  <Group>
    <InputFolder path="C:\Scan Server\CompressTiff\Input" filter="*.tif"/>
    <OutputFolder path="C:\Scan Server\CompressTiff\Output"/>
    <FailedFolder path="C:\Scan Server\CompressTiff\Failed"/>
  </Group>
  <Processing>
    <Tiff.SplitPages/>
    <List.Map>
      <Tiff.Compress upgrade-jpeg="true" recompress-lossy="false">
        <Bilevel-Compression type="group4"/>
        <Grayscale-Compression type="jpeg" quality="90"/>
        <RGB-Compression type="jpeg" quality="75"/>
      </Tiff.Compress>
    </List.Map>
    <Tiff.Merge/>
  </Processing>
  </Group>
  <!-- Create mixed raster content PDF from TIFF
  =============== -->
  <Group>
    <InputFolder path="C:\Scan Server\Tiff2MrCPdf\Input" filter="*.tif"/>
    <OutputFolder path="C:\Scan Server\Tiff2MrCPdf\Output"/>
    <FailedFolder path="C:\Scan Server\Tiff2MrCPdf\Failed"/>
  </Group>
  <Processing>
    <Tiff.SplitPages/>
    <List.Map>
      <Tiff.Ocr plugin="service" languages="German,English"/>
    </List.Map>
  </Processing>
</ScanServer>
```
Convert JPEG image to compressed TIFF

- Since JPEG doesn't allow multiple pages,
- there is no per-page parallelization in this case

```xml
<Tiff.ImportImage />
<Tiff.Compress upgrade-jpeg="true" recompress-lossy="false">
  <Bilevel-Compression type="group4"/>
  <Grayscale-Compression type="jpeg" quality="90"/>
  <RGB-Compression type="jpeg" quality="75"/>
  <MRC segmentation="true">
    <MaskFill-Compression quality="20" scale="2"/>
  </MRC>
</Tiff.Compress>
</Group>
</List.Map>
<Tiff.ConvertToPdf/>
</Processing>
</Group>
</ScanServer>
```
6 Cook book

6.1 Splitting a TIFF by barcode

Whenever a specific barcode containing the text "SpLiT" appears on a page, the document is split into parts.

Configuration

```xml
<Processing>
    <Tiff.SplitPages/>
    <List.ChunkIf>
        <Tiff.Ocr plugin="service"
                  parameters="PredefinedProfile=BarcodeRecognition_Accuracy"
                  max-parallel="4"/>
        <Tiff.ExtractOcrData/>
        <List.ElementAt index="0"/>
        <Xml.Parse/>
        <XPath.Select xmlns:ocr="http://www.pdf-tools.com/ocr">
            //ocr:barcode[text()="SpLiT"]
        </XPath.Select>
        <List.IsEmpty/>
        <Bool.Not/>
    </List.ChunkIf>
    <List.Map>
        <Tiff.Merge/>
    </List.Map>
    <FileList.AppendSuffix>_{0}</FileList.AppendSuffix>
</Processing>
```

Explaining the details

1. The multi page TIFF file is split into a list of single page TIFFs.
2. For each of the single page TIFFs it is determined, if it contains the required barcode by:
   a. Perform OCR on the TIFF using a profile suited for barcode recognition. The recognized content is embedded into the TIFF.
   b. The recognized content is extracted from the TIFF.
   c. Since a TIFF can potentially contain multiple pages, we only use the content of the first page (we know that it's only one page in this context)
   d. With an XPath expression, a list of barcode elements is extracted from the content.
   e. If this list is not empty, that means that the page contains the required barcode.
3. The list of single page TIFFs is split into sub-lists, whenever a page contains the required subpage.
4. Each sublist is again merged into a multi page TIFF.
5. The files are renamed, such that the filenames contain an index.
6.2 Merge a folder of TIFF files into a single PDF

Scan the input folder for subfolders that are processed as a whole as soon as a text file is dropped into the subfolder. The conversion includes OCR processing and compression.

Configuration

```xml
<ScanServer>
  <Workfolder path="C:\ScanServer\Workfolder"/>
  <Group>
    <InputFolder path="C:\ScanServer\Input" folders="true"
                 trigger="*.txt"/>
    <OutputFolder path="C:\ScanServer\Output"/>
    <FailedFolder path="C:\ScanServer\Failed"/>
    <Processing>
      <Folder.Files filter="*.tif"/>
      <List.SortBy comparison="logical">
        <File.Name/>
      </List.SortBy>
      <List.Map>
        <Tiff.SplitPages/>
        <List.Map>
          <Tiff.Ocr plugin="service@http://localhost:7982"/>
          <Tiff.Compress>
            <Bilevel-Compression type="Group4"/>
            <Grayscale-Compression type="JPEG" quality="75"/>
            <RGB-Compression type="JPEG" quality="75"/>
          </Tiff.Compress>
        </List.Map>
      </List.Map>
    </Processing>
  </Group>
</ScanServer>
```

Explaining the details

1. The input folder is scanned for folders, the processing of a subfolder is started as soon as a .txt file is found in the subfolder.
2. The files with extension .tif are extracted from the subfolder.
3. The list of files is sorted by name, using the logical ordering (same as Windows Explorer).
4. Each file is processed separately:
   a. The file is split into one TIFF file per page and each page is again processed separately:
      i. OCR is performed using the OCR service on the same machine.
      ii. The page is compressed using different compression algorithms for different image types.
5. The nested list is joined into a single list of TIFF files (one per page).
6. The list of TIFFs is then converted into a single PDF.
# Version History

## Changes in Version 6

- **Improved** search algorithm for installed fonts: User fonts under Windows are now also taken into account.

## Changes in Version 5

- **New** additional supported operating system: Windows Server 2019.
- **Changed** behavior when reading a TIFF. The value `Relative` from tag `ResolutionUnit` is now interpreted as `Inch`.

## Changes in Version 4.12

- **New** HTTP proxy setting in the GUI license manager.
- **Introduced** license features `TiffProcessing` and `PdfSignature`.

### Component `Tiff.ExtractOcrData`

- **New** attribute `format-version` to produce a fixed version of the OCR XML format.

## Changes in Version 4.11

- **New** support for reading and writing PDF 2.0 documents.
- **New** support for the creation of output files larger than 10GB (not PDF/A-1).
- **Improved** search in installed font collection to also find fonts by other names than TrueType or PostScript names.
- **Improved** font subsetting of CFF and OpenType fonts.

### Component `ScanServer.Root`

- **New** attribute `lock-files` on `<InputFolder>` element to configure whether input files are locked during processing.
  - **Changed** the default value from "true" to "false".

### Component `Tiff.ExtractOcrData`

- **Changed** OCR XML format (version 3)

## Changes in Version 4.10

### Component `Tiff.Merge`

- **New** attribute `remove-blank-pages="true"` to remove pages without content.
7.6 Changes in Version 4.9

- **New** support for OpenType font collections in installed font collection.
- **Improved** metadata generation for standard PDF properties.

**Component Tiff.ConvertToPdf**

- **New** attribute `xmp-path` to add XMP metadata to the PDF.
- **New** attribute `fallback-color-profile-path` to specify a color profile for TIFFs that have none embedded.

**Component Folder.Files**

- **Changed** attribute `filter`: The attribute now accepts multiple filter strings separated by semicolon.
- **New** attribute `regex` to filter filenames by a regular expression.

7.7 Changes in Version 4.8

- **New** feature: Merge files from an input subfolder
- **New** component: `<List.SortBy>` to sort a list of items.
- **New** component: `<List.Join>` to flatten nested lists of items.
- **New** component: `<File.Name>` to get the name of a file.
- **New** component: `<Folder.Files>` to extract the content of a folder.

**Component ScanServer.Group**

- **New** attribute `<InputFolder folders="...">` to scan the input folder for subfolders.
- **New** attribute `<InputFolder trigger="...">` to specify a trigger file.
8 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** The 3-Heights™ Scan to PDF Server is copyrighted. This user’s manual is also copyright protected; it may be copied and given away provided that it remains unchanged including the copyright notice.

**Contact**

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

A.1 Versions

The XML format has evolved over time and will continue to do so. Incompatible changes are denoted by increasing the format version. The current version is 3.

If applicable, the minimum format version of attributes or child elements is specified in parentheses.

The addition of new optional attributes is not considered an incompatible change. Applications that consume the XML must therefore be prepared to ignore unknown attributes.

A.2 Elements

A.2.1 <document> Element

The root element of the XML.

This element is omitted if the XML is describing a single page only. In that case, the <page> element is the root element.

Attributes:

version (optional in v1, required otherwise) The version of the XML format.

Default value is "1".

Child elements:

<page> (v1, 1..n)

A.2.2 <page> Element

A single page that represents the recognized image.

Attributes:

version (optional) The version of the XML format.

If no version is specified, the value is inherited from the parent <document> element, if present. Default value is "1".

bb The bounding box of the page in pixels: "0 0 w h"

res The resolution of the image.

Child elements:

&page-content> (v1, 1)

A.2.3 <page-content> Element

The root element of the page content.

Attributes:

tf (optional), font-name (optional), font-family (optional), font-styles (optional), font-size (optional), locale (optional)
A.2.4 <division> Element

A generic group of content elements.

Attributes:
- tf (optional), font-name (optional), font-family (optional), font-styles (optional), font-size (optional), locale (optional)

Child elements:
- <text> (v1, 0..n), <division> (v3, 0..n), <heading> (v3, 0..n), <paragraph> (v3, 0..n), <table> (v3, 0..n), <image> (v3, 0..n), <barcode> (v3, 0..n)

A.2.5 <heading> Element

A text heading.

Attributes:
- tf (optional), bb (optional), font-name (optional), font-family (optional), font-styles (optional), font-size (optional), locale (optional)

Child elements:
- <text> (v3, 0..n), <division> (v3, 0..n), <heading> (v3, 0..n), <paragraph> (v3, 0..n), <table> (v3, 0..n), <image> (v3, 0..n), <barcode> (v3, 0..n)

A.2.6 <paragraph> Element

A text paragraph.

Attributes:
- tf (optional), bb (optional), font-name (optional), font-family (optional), font-styles (optional), font-size (optional), locale (optional)

Child elements:
- <text> (0..n)

A.2.7 <text> Element

A text fragment.

The base line of the text is determined by the line $y = 0$ in the transformed coordinate system. Usually, the transformation looks like $tf=\left[\begin{array}{cc}1 & 0 \\ 0 & 1 \end{array}\right] \times x$, where $(x, y)$ is the baseline position of the first character.

The baseline and the bounding box are not necessarily intersecting.

Note: In version 1 of the format, barcodes are represented by <text> elements with font-name="Barcode" or font-name="BarcodeHex".

Attributes:
- tf (required), bb (required), font-name (optional), font-family (optional), font-styles (optional), font-size (optional), locale (optional),
suspicious-chars  (optional) The zero based indexes of all suspicious characters, separated by a space. If the attribute is empty, there are no suspicious characters.
If the attribute is missing, the information is unknown.

char-left-pos    (optional) The position of the left border of each character, separated by a space. The position is measured on the baseline starting from the left border.
If the attribute is missing, the information is unknown.

char-right-pos   (optional) The position of the right border of each character, separated by a space. The position is measured on the baseline starting from the left border.
If the attribute is missing, the information is unknown.

Text content:
The text content of the text fragment as plain text.

A.2.8 <table> Element
A table.

Attributes:  tf (optional), font-name (optional), font-family (optional), font-styles (optional), font-size (optional), locale (optional)

Child elements:
<row> (0..n)

A.2.9 <row> Element
A table row.

Attributes: font-name (optional), font-family (optional), font-styles (optional), font-size (optional), locale (optional)

Child elements:
<cell> (0..n)

A.2.10 <cell> Element
A table cell.

Attributes:
font-name (optional), font-family (optional), font-styles (optional), font-size (optional), locale (optional)

type  (optional) The cell type.
"data"  (Default) A data cell.
"heading"  A header cell.

Child elements:
<text> (v3, 0..n), <division> (v3, 0..n), <heading> (v3, 0..n), <paragraph> (v3, 0..n), <table> (v3, 0..n),
<image> (v3, 0..n), <barcode> (v3, 0..n)
A.2.11 〈image〉 Element

An image.

Attributes:
- **tf** (optional),  
- **bb** (optional),  
- **type** (optional) The image type.
  - "raster" (Default) A raster picture.  
  - "vector" A vector graphic.

A.2.12 〈barcode〉 Element

An 1D or 2D barcode.

Note: In version 1 of the format, barcodes are represented by 〈text〉 elements with font-name="Barcode" or font-name="BarcodeHex".

Attributes:
- **tf** (optional),  
- **bb** (optional),  
- **encoding** (optional) The encoding of the barcode value.
  - "hex" Encoded as a hexadecimal string.

Text content:
The barcode value as plain text or in the specified encoding.

A.3 Common Attributes

A.3.1 tf Attribute

The coordinate transformation matrix of the element.

**tf**="m11 m12 m21 m22 dx dy"

The transformation matrix is specified by six numbers. All information about orientation, rotation, scaling, skewing and translation can be calculated based on these six numbers.

The actual matrix is $$M = \begin{bmatrix} m_{11} & m_{12} & 0 \\ m_{21} & m_{22} & 0 \\ dx & dy & 1 \end{bmatrix}$$

This matrix is used to define a transformation of a vector [x y 0] to a vector [x’ y’ 0] = [x y 0] · M, where (x, y) is the original point and (x’, y’) is the transformed point on the page.

A.3.2 bb Attribute

The bounding box of the element in the transformed coordinate system.

**bb**="x y w h"

If the element also contains the **tf** attribute, the transformation is applied, before the bounding box is computed.
A.3.3 font-name Attribute

The name of the font used for `<text>` elements.

The attribute is inheritable, i.e. it can occur on any parent element of the text, down to the `<page-content>`.

A.3.4 font-family Attribute

The family of the font used for `<text>` elements, separated by a space.

Possible values are:

"mono" A monospaced font, i.e. every character has the same width. An example of such a font is "Courier".
"sans" A font without serifs (sans serif). An example of such a font is "Arial".
"serif" A font with serifs. An example of such a font is "Times".

The attribute is inheritable, i.e. it can occur on any parent element of the text, down to the `<page-content>`.

A.3.5 font-styles Attribute

The list of styles of the font used for `<text>` elements, separated by a space.

Possible values are:

"bold"
"italic"
"underline"
"strikeout"

The attribute is inheritable, i.e. it can occur on any parent element of the text, down to the `<page-content>`.

A.3.6 font-size Attribute

The size the font used for `<text>` elements.

The attribute is inheritable, i.e. it can occur on any parent element of the text, down to the `<page-content>`.

A.3.7 locale Attribute

The locale of `<text>` elements in ISO format.

The attribute is inheritable, i.e. it can occur on any parent element of the text, down to the `<page-content>`.
A.4 Example

<?xml version="1.0" encoding="utf-8"?>
<page xmlns="http://www.pdf-tools.com/ocr" version="3" bb="0 0 2481 3508" res="300 300">
  <page-content font-name="Times New Roman" font-family="serif" font-styles=""
                font-size="13" locale="en-US">
    <div font-styles="#b" font-size="18">
      <heading bb="297 314 1879 381">
        <text tf="1 0 0 1 297 366" bb="0 -51 146 0" suspicious-chars="" char-left-pos="0 48 84 118" char-right-pos="41 80 112 146">Face</text>
      </heading>
      <paragraph bb="296 623 2110 727" font-size="12" font-styles="">
        <text tf="1 0 0 1 430 658" bb="0 -33 122 11" suspicious-chars="" char-left-pos="0 16 55 78 103" char-right-pos="14 54 74 100 122">Image</text>
        <text tf="1 0 0 1 568 658" bb="0 -35 87 11" suspicious-chars="" char-left-pos="0 25 45 71" char-right-pos="23 44 70 87">days</text>
      </paragraph>
    </div>
    <table font-name="Arial" font-family="sans" font-size="10">
      <row>
        <cell>
          <paragraph bb="299 1042 458 1071">
            <text tf="1 0 0 1 299 1071" bb="0 -29 76 0" suspicious-chars="1 2" char-left-pos="0 20 32 46 63" char-right-pos="15 24 43 60 76">First</text>
          </paragraph>
        </cell>
        <cell>
          <paragraph bb="935 1040 1138 1071">
            <text tf="1 0 0 1 935 1071" bb="0 -29 78 0" suspicious-chars="" char-left-pos="0 25 58" char-right-pos="22 54 78">Two</text>
          </paragraph>
        </cell>
      </row>
      <row>
        <cell>
          <paragraph bb="297 1008 513 1129">
            <text tf="1 0 0 1 297 1129" bb="0 -31 131 0" suspicious-chars="" char-left-pos="0 21 44 63 89 112" char-right-pos="17 40 60 83 107 131">Second</text>
          </paragraph>
        </cell>
        <cell>
          <paragraph bb="937 1098 1217 1129">
            <text tf="1 0 0 1 937 1129" bb="0 -31 131 0" suspicious-chars="" char-left-pos="0 21 44 64 89 112" char-right-pos="17 40 60 84 107 131">Second</text>
          </paragraph>
        </cell>
      </row>
    </table>
  </page-content>
</page>