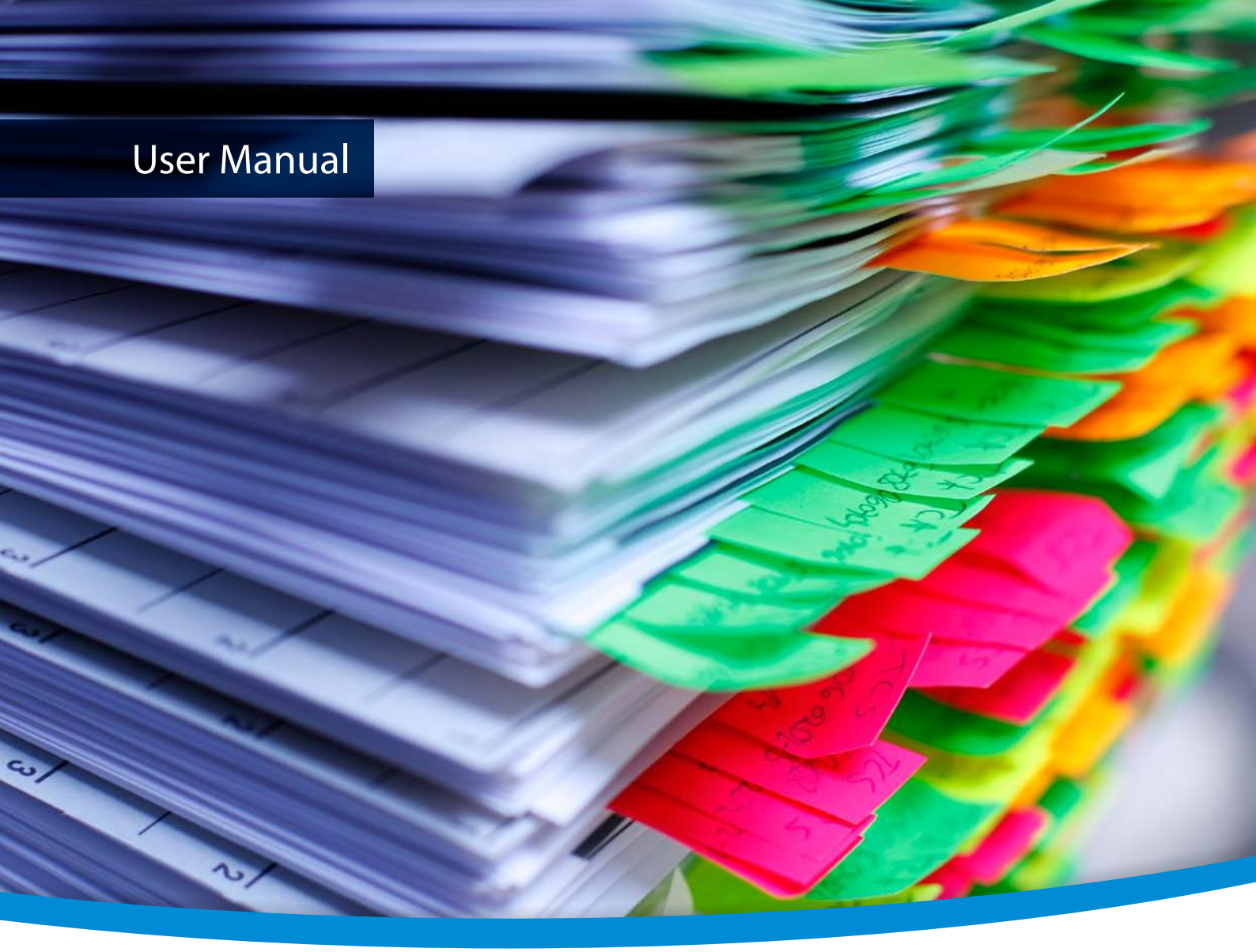
User Manual



3-Heights®
Document Converter API

PDF/A conform

Version 6.24.1

PDF-TOOLS.COM
Premium PDF Technology

Contents

1 Introduction

This documentation provides additional information about how to use the API of the 3-Heights® Document Converter Enterprise and SME edition.

The 3-Heights® Document Converter Enterprise documentation and SME documentation can be found on the Pdftools website.

1.1 Interfaces

The following interfaces are available:

- C
- COM
- .Net

1.2 Operating systems

The 3-Heights® Document Converter API is available for the following operating systems:

- Windows Client 7+ | x64
- Windows Server 2008 R2, 2012, 2012 R2, 2016, 2019, 2022 | x64

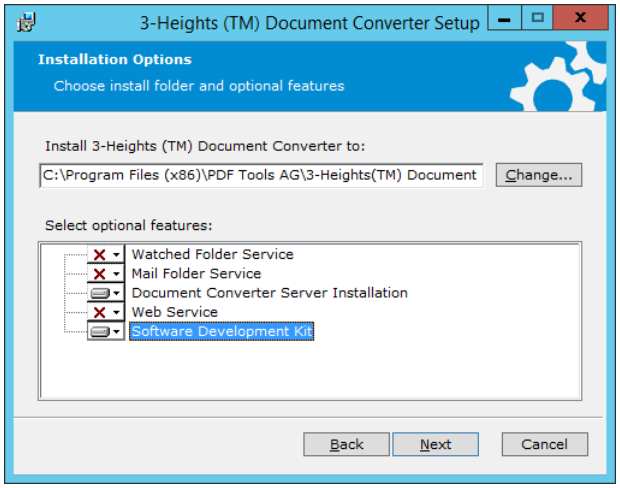
'+' indicates the minimum supported version.

2 Installation

The Document Converter API is installed as part of the 3-Heights® Document Converter Enterprise or SME Service installation.

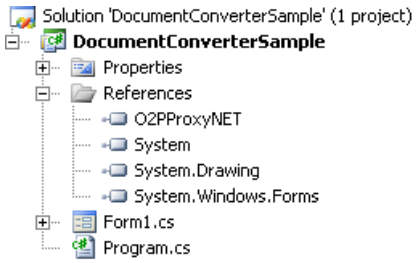
Therefore execute the Microsoft Installer package `DocumentConverterEnterprise-‹version›-Windows-(‹platform›).msi`.

During this installation, select the feature “Software Development Kit”, as indicated in the screenshot below.



This installs the `O2PProxyAPI.DLL` into the installation directory, which incorporates programming interfaces for COM and C.

When using .NET add `O2PProxyNET.dll` as a .NET reference to your project.



For simplicity add the namespace `Pdftools.Converter` to your project.

```
using Pdftools.Converter;
```

3 Conversion job

The purpose of this chapter is to elaborate the different steps of a conversion job for the available interfaces.

3.1 Initialization

COM:

```
Set oConverter = CreateObject("O2PConverter.IConverter")
```

.NET:

```
using Pdftools.Converter;
IConverterService converter = ConverterFactory.GetInstance(null);
```

C:

```
#include "o2pproxyapi_c.h"
O2PConverter hConverter = O2PCreateConverter(0);
```

During creation of the Converter object, the API attempts to connect to the Document Converter. The location URL can be passed to the API when using the .NET or C interfaces. When a null pointer is specified (or during construction of the COM object), the API tries to retrieve the URL from the application's configuration file. If this fails, the default URL is used (`tcp://localhost:7981/O2Pservice`).

De-initialization is performed implicitly with the .NET and COM interfaces when the last reference to the object is released. When using the C interface, the `O2PDestroyConverter` function is called.

3.2 Job creation

COM:

```
Set oJob = oConverter.CreateJob
```

.NET:

```
IJob job = Converter.CreateJob();
```

C:

```
O2PJob hJob = O2PConverterCreateJob(hConverter);
```

Prior to perform any job related processing, the application obtains a job object as shown above.

3.3 Setting job options

COM:

```
oJob.SetOptions("PDFA")
```

.NET:

```
job.SetOptions("PDFA");
```

C:

```
O2PJobSetOptions(hJob, "PDFA");
```

Job options can be set any time and will replace any options that have been set previously.

The options string is composed of a sequence of semicolon separated key-value pairs, where key and value are separated by an equal sign. See section Job Options in the Document Converter Enterprise or SME documentation, for a detailed list of available options.

> **Note:** In the sample code above, the option key is `"PDFA"` whose value can be either `true` or `false`. For setting a `true` value, it is sufficient to just list the key value.
> The terminating semicolon is not necessary.

3.4 Document conversion job

Conversion jobs begin with `CreateOutput`, followed by `AppendDoc` calls prior to the `Close` call. It is possible to pass options with the `AppendDoc` method, those options are called document options. See section Document Options in the Document Converter Enterprise or SME documentation.

COM:

```
bDone = oJob.CreateOutput("C:\Temp\result.pdf")
bDone = oJob.AppendDoc("a.doc")
bDone = oJob.AppendDoc("b.xls")
...
bDone = oJob.Close()
```

.NET:

```
job.CreateOutput("result.pdf");
FileStream fs = File.Open("a.doc", FileMode.Open, FileAccess.Read,
FileShare.Read);
byte[] documentContent = new byte[fs.Length];
fs.Read(documentContent, 0, fs.Length); fs.Close();
job.AppendDoc(documentContent, "Outline=First Document")
...
job.Close()
```

C:

```
O2PJobCreateOutput(hJob, "result.pdf");
O2PJobAppendDoc(hJob, "a.doc", "");
...
O2PJobClose(hJob);
```

`CreateOutput` and **`AppendDoc`** return a Boolean value. A `False` value signals an error situation.

`JobClose` Returns the number of pages converted. The application should always verify at this point if there are any warnings reported.

3.5 Obtain information about warnings or errors

The API contains functions to return an error code and an error text:

COM:

```
oJob.ErrorCode
oJob.ErrorText
```

.NET:

```
ErrorInfo e = Job.GetLastError()
Console.WriteLine("Error code: " + e.ErrorCode);
Console.WriteLine("Error text: " + e.ErrorText);
```

C:

```
int iErrorCode;
TCHAR* pszErrorText;
O2PJobGetLastError(hJob, &iErrorCode, &pszErrorText);
```

Errors returned may originate from the Document Converter API, but could also be propagated from system calls.

The Document Converter API specific errors are listed in the section Error codes

3.6 Passing parameter data for custom plugins

The `AppendDoc` method accepts optional parameter data which is transported along with the document data. This can be useful for document conversion by a custom plugin.

Parameter data consists of an array of key/value pairs. A key is a name string that permits identification and retrieval of the value data, which is passed as a byte array.

A server side plugin will thus be able to retrieve the parameter data when its “OnDocumentOpen” method is activated.

To facilitate passing of parameter date with the `o2pclient.exe` command line tool, the `O2PProxyAPI.dll` supports passing of parameter data by a special treatment of document options identified by a leading hash (#) sign. When specifying `#Data=path-to-parameter-data` as the first document option, the API will read the contents of the file with the specified name and pass this as parameter data with name “Data”. The option is removed from the beginning of the document options string before being forwarded to the server.

4 Interface reference

This chapter describes all functions of the Document Converter API. Sample code is provided in C#. If you are not familiar with the Document Converter, it may be easiest to first have a look at the samples provided in the chapter Examples.

4.1 IConverterService Interface

An instance of an IConverterService is created using the class Class ConverterFactory.

4.1.1 CreateJob

Method: `IJob CreateJob()`

Create a conversion job and return its interface.

4.1.2 GetKnownFileExtensions

Method: `String GetKnownFileExtensions()`

Retrieve the file extensions that are known to be supported by the Document Converter.

Returns:

A colon separated list of file extensions

Example:

```
"doc:docx:xls:xlsx:ppt:pptx:pdf:bmp:gif:jpg:jpeg:png"
```

4.1.3 GetLastError

Method: `ErrorInfo GetLastError()`

Get the last error code.

4.1.4 ProcessConversion

Method: `string ProcessConversion(string OfficeDocumentPath, string Options, Parameter[] Params, string ResultPdfPath)`

Convert a single document. This function is performed outside of the Job environment, which means that no Job related processing such as OCR, digital signing, or PDF/A conversion and validation is performed.

Parameters:

`OfficeDocumentPath` `[string]` The file system path of the input document.

`Options` `[string]` The processing options.

`Params` `[Parameter[]]` Any parameters to be forwarded to plugins.

`ResultPdfPath` `[string]` The file system path where the output shall be saved to.

Returns:

The document title extracted from the document.

4.2 Class ConverterFactory

The `GetInstance` method of the `ConverterFactory` class is used to obtain a `IConverterService` object.

4.2.1 GetInstance

Method: `public static GetInstance(string ServicePoint)`

Returns:

The IConverterService interface using the ServicePoint string to establish a remote connection to the Document Converter Service.

This method may throw .NET remoting exceptions, specifically a System.Net.Sockets.SocketException when the remote connection fails.

Example:

```
IConverterService converter;
converter = ConverterFactory.GetInstance("tcp://localhost:7981/O2PService");
```

Where localhost should be replaced by the real name of the server where the Document Converter resides.

4.3 IJob Interface

This interface controls one thread of document conversion and can be reused for multiple sequential conversions.

4.3.1 AppendDoc

Method: `bool AppendDoc(byte[] DocumentBytes, string Options)`

Method: `bool AppendDoc(string OfficeDocumentPath, string Options)`

Method: `bool AppendDoc(byte[] DocumentBytes, string Options, Parameter[] Params)`

Add a document to the job for conversion.

Parameters:

`DocumentBytes` `[byte[]]` The document can be passed as by reference (OfficeDocumentPath) or by value (DocumentBytes). In case the document is passed by file name, the client application and the Document Converter server must have a shared file system, e.g. are on the same host.

`Options` `[string]` Any options to be passed to the conversion process (e.g. open password, page format, etc.). For a full list of supported options, please see section "Document Options" in the Enterprise or SME documentation.

`Params` `[Parameter[]]` Additional parameter data.

Returns:

`True` On success.

`False` Otherwise.

4.3.2 Cancel

Method: `void Cancel()`

Cancel the job, so it can be re-used for a new conversion.

4.3.3 CreateOutput

Method: `bool CreateOutput(string Filename)`

This function initializes a new conversion job.

This function must be called before documents are converted and appended using AppendDoc.

Parameter:

Filename [string] To create a document on the file system, provide a valid file name for the target output document. In order to create a document in memory, provide null as parameter.

Returns:

True If a document was created successfully.

False Otherwise.

Example: Create document on file system

```
job.CreateOutput("@C:\temp\output.pdf");
```

Create document in memory:

```
job.CreateOutput(null);
```

4.3.4 FinishConversion

Method: `int FinishConversion()`

Complete any pending conversion processing and close the result PDF.

Use this function to define the end of the conversion, i.e. no more documents are appended to the current job.

Returns:

The number of converted pages.

4.3.5 GetLastError

Method: `ErrorInfo GetLastError()`

This function provides error information when one of the other calls in this interface indications failure by returning a `false` value.

Returns:

It returns an `ErrorInfo` structure consisting of a code and a text.

4.3.6 RetrieveMeta

Method: `byte[] RetrieveMeta()`

Retrieve any meta information gathered during conversion processing.

> **Note:** For future use; returns `null` in the current version.

Returns:

The meta information is an 8 bit ASCII buffer.

4.3.7 RetrieveOutput

Method: `byte[] RetrieveOutput()`

Retrieve the output PDF once the job is completed. Must be called after `FinishConversion`.

If the conversion was successful, the document is returned as a byte array. This function can only be used if the document was created in memory, see `CreateOutput`. How the function is used, see Remote, by value.

4.3.8 SetDocMetadata

Method: `bool SetDocMetadata(byte[] XmpMetadata)`

Set the XMP Metadata stream to be included in the PDF output document. If no metadata is set, default metadata is created by the Document Converter.

4.3.9 SetOptions

Method: `bool SetOptions(string Options)`

Set job options.

Parameter:

`Options` `[string]` The options string is composed of a sequence of semicolon separated key-value pairs, where key and value are separated by an equal sign. For setting a true value, it is sufficient to just list the key value.

For a full list of supported options, please see section "Job Options" in the Enterprise or SME documentation.

Returns:

`True` If options are valid

`False` Otherwise.

Example:

```
job.SetOptions("PDFA;CONVERTALWAYS=true");
```

4.3.10 Terminate

Method: `void Terminate()`

This function frees all resources and disconnects the remote connection.

4.4 Struct ErrorInfo

4.4.1 ErrorCode

Method: `public uint ErrorCode()`

The error code as uint. Error codes are described in section Error codes.

4.4.2 ErrorText

Method: `public string ErrorText()`

A descriptive English text of the error code.

4.4.3 ErrorInfo

Method: `public ErrorInfo(uint c, string s)`

This is the constructor for storing an error code and the description string.

4.5 Error codes

The 3-Heights® Document Converter API specific errors are listed in the following table.

Error codes

Error Code and Message	Background
`O2P_E_UNKFORMAT = 0x82410C01` `The file has an unknown format`	The input document format could not be recognized. The document may be corrupted.
`O2P_E_INVALIDOP = 0x82410C02` `Invalid state for requested operation`	An inappropriate API call was made (e.g. wrong call sequence).
`O2P_E_NOPAGES = 0x82410C03` `The document contains no pages`	The document to be converted does not contain any pages.
`O2P_E_NOPDFPRINTER = 0x82410C04` `No suitable PDF Printer installed`	The PDF Producer printer entries are not available or have been deleted.
`O2P_E_NOSCREENS = 0x82410C05` `No screen session available`	Terminal server sessions are not available.
`O2P_E_PRINTTIMEOUT = 0x82410C06` `Printing timeout experienced`	The output from printing is overdue. The log file may contain more information.
`O2P_E_SVCUNAVAIL = 0x82410C07` `Document Converter unavailable`	The client failed to connect to the (remote) server. It may not have been started yet, or the URI may not be configured correctly.
`O2P_W_PARTSMISSING = 0x02410C08` `Some parts of the document could not be processed`	A compound document (mail with attachments, ZIP archive) contains elements that could not be converted.
`O2P_E_PDFACONVFAIL = 0x82410C09` `PDF/A conversion failed`	Failure may be due to transparency in the input document, missing color profiles, or other issues. See log files.
`O2P_E_UNKNOWN = 0x82410C0A` `Generic error`	The log file may contain more information.
`O2P_E_APPERROR = 0x82410C0B` `Print application specific error`	MS Word or some other third party application has encountered a problem. See log files.
`O2P_W_PDFACONVWARN = 0x02410C0C` `PDF/A conversion completed with warnings`	See log files or error description returned for more detailed information.
`O2P_E_SOURCECORRUPT = 0x82410C0D` `Source document validation failed`	An input document is not conforming to validation criteria.
`O2P_W_SOURCEQUALITY = 0x02410C0D` `Source document inconsistencies detected`	An input document contains invalid links or bookmarks or is not a conform Office document.

Error codes

O2P_E_PASSWORD = 0x82410C0E Password required	The input document is password protected, or the specified password is incorrect.
O2P_W_DECRYPTERROR = 0x02410C0E	Content decryption error (mail)
O2P_E_OCR = 0x82410C0F OCR error (engine not available)	Verify that the specified OCR engine is available on the server.
O2P_E_PDFACOMPLIANCE = 0x82410C10 PDF/A conversion failed to reach required conformance level	Structure information may be missing in the input document, preventing conversion to PDF/A-1a.
O2P_I_VOIDRESULT = 0x02410C11 Conversion was successful, result is empty	This code can be returned by a plugin, indicating that it will handle further processing of the output.
O2P_E_PLUGINERROR = 0x82410C12 An unhandled error occurred in a plugin	The log file may contain more information.
O2P_E_VBAERROR = 0x82410C13 VBA error in document	The document contains a macro that caused an error.
O2P_E_APPBLOCKED = 0x82410C14 office application was blocked or failed	The application used to convert a document crashed or was blocked.
O2P_W_SIGN = 0x02410C15 document could not be signed	An error occurred while trying to sign the document.
O2P_E_POPUPBLOCKING = 0x82410C16	Application pop-up dialog box cannot be closed
O2P_W_OPTIMIZE = 0x02410C17	PDF optimization failed
O2P_E_HTTPSTATUS = 0x82410C18	HTTP error while accessing source document.
O2P_W_DOWNGRADE = 0x02410C19 Downgrade during PDF/A conversion	Warning if conformance downgraded during PDF/A conversion.
O2P_W_INPUTSIGNED = 0x02410C1A	Input document is signed (warning)
O2P_E_INPUTSIGNED = 0x82410C1A	Input document is signed (error)
O2P_E_INPUTCORRUPT = 0x82410C1B	Input document corruptions detected
O2P_E_RENDERINGFAILED = 0x82410C1C	Rendering of document failed (e.g. XFA rending via Acrobat Reader)
O2P_W_NOTPDFA = 0x02410C1D	Result is not PDF/A
O2P_W_UPGRADE = 0x02410C1E	PDF/A conformance was upgraded to PDF/A-2

Other error codes originating from Windows can also be returned. As an example, a code of 2 may be returned from the Windows file system (The system cannot find the file specified).

> **Note:**
> - It is important to note that PDF/A conversion, OCR recognition and digital signing is performed at the end of job processing (i.e. during `JobClose`).
> - Errors related to OCR and digital signing are often due to configuration issues, while PDF/A conversion problems become manifest due to the specific input.

Common reasons for PDF/A conversion to fail

- Invalid/corrupted XMP Metadata.
- No suitable font is available to be embedded.
- The document contains transparency features.
- The document contains layers (optional content feature).
- Appearance objects of annotations are missing.
- Unknown or prohibited annotation types.
- Embedded files.

Possible problems may also arise, if the original document contains Javascript resources or digital Signatures. Javascripts are removed from the document, which normally should not have any negative effects.

Digital signatures however are inherently invalidated by the PDF/A conversion process. It may thus make sense to remove such signatures prior to passing any PDF files for conversion.

5 Web service

5.1 WSDL definition

The web service interface features methods to convert a document (or ZIP package of documents) in a single call. Here is the WSDL definition:

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions xmlns:s="http://www.w3.org/2001/XMLSchema"
                  xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
                  xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
                  xmlns:tns="http://pdf-tools.com/ws/o2p/"
                  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
                  xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
                  xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
                  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
                  targetNamespace="http://pdf-tools.com/ws/o2p/"
                  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
    <s:schema elementFormDefault="qualified" targetNamespace=
              "http://pdf-tools.com/ws/o2p/">
      <s:element name="ConvertFile">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="DocBytes"
                       type="s:base64Binary" />
            <s:element minOccurs="0" maxOccurs="1" name="Options"
                       type="s:string" />
            <s:element minOccurs="0" maxOccurs="1" name="Metadata"
                       type="s:base64Binary" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="ConvertFileResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="1" maxOccurs="1" name="ConvertFileResult"
                       type="tns:ConversionResult" />
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:complexType name="ConversionResult">
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="1" name="DocumentBytes"
                     type="s:base64Binary" />
          <s:element minOccurs="1" maxOccurs="1" name="NumPages"
                     type="s:int" />
          <s:element minOccurs="1" maxOccurs="1" name="ErrorCode"
                     type="s:int" />
          <s:element minOccurs="0" maxOccurs="1" name="ErrorDescription"
                     type="s:string" />
        </s:sequence>
      </s:complexType>
      <s:element name="ConvertFile2">
        <s:complexType>
```

```
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="1" name="InputPath"
                     type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="Options"
                     type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="Metadata"
                     type="s:base64Binary" />
          <s:element minOccurs="0" maxOccurs="1" name="OutputPath"
                     type="s:string" />
        </s:sequence>
      </s:complexType>
    </s:element>
    <s:element name="ConvertFile2Response">
      <s:complexType>
        <s:sequence>
          <s:element minOccurs="1" maxOccurs="1" name="ConvertFile2Result"
                     type="tns:ConversionResult" />
        </s:sequence>
      </s:complexType>
    </s:element>
    <s:element name="ConvertFileWithData">
      <s:complexType>
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="1" name="DocBytes"
                     type="s:base64Binary" />
          <s:element minOccurs="0" maxOccurs="1" name="Options"
                     type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="Metadata"
                     type="s:base64Binary" />
          <s:element minOccurs="0" maxOccurs="1" name="ParameterName"
                     type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="ParameterData"
                     type="s:base64Binary" />
        </s:sequence>
      </s:complexType>
    </s:element>
    <s:element name="ConvertFileWithDataResponse">
      <s:complexType>
        <s:sequence>
          <s:element minOccurs="1" maxOccurs="1"
                     name="ConvertFileWithDataResult"
                     type="tns:ConversionResult" />
        </s:sequence>
      </s:complexType>
    </s:element>
    <s:element name="ConvertUrl">
      <s:complexType>
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="1" name="Url"
                     type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="Options"
                     type="s:string" />
          <s:element minOccurs="0" maxOccurs="1" name="MetadataUrl"
                     type="s:string" />
        </s:sequence>
      </s:complexType>
    </s:element>
    <s:element name="ConvertUrlResponse">
```

```
      <s:complexType>
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="1" name="ConvertUrlResult"
                     type="s:string" />
        </s:sequence>
      </s:complexType>
    </s:element>
  </s:schema>
</wsdl:types>
<wsdl:message name="ConvertFileSoapIn">
  <wsdl:part name="parameters" element="tns:ConvertFile" />
</wsdl:message>
<wsdl:message name="ConvertFileSoapOut">
  <wsdl:part name="parameters" element="tns:ConvertFileResponse" />
</wsdl:message>
<wsdl:message name="ConvertFile2SoapIn">
  <wsdl:part name="parameters" element="tns:ConvertFile2" />
</wsdl:message>
<wsdl:message name="ConvertFile2SoapOut">
  <wsdl:part name="parameters" element="tns:ConvertFile2Response" />
</wsdl:message>
<wsdl:message name="ConvertFileWithDataSoapIn">
  <wsdl:part name="parameters" element="tns:ConvertFileWithData" />
</wsdl:message>
<wsdl:message name="ConvertFileWithDataSoapOut">
  <wsdl:part name="parameters" element="tns:ConvertFileWithDataResponse" />
</wsdl:message>
<wsdl:message name="ConvertUrlSoapIn">
  <wsdl:part name="parameters" element="tns:ConvertUrl" />
</wsdl:message>
<wsdl:message name="ConvertUrlSoapOut">
  <wsdl:part name="parameters" element="tns:ConvertUrlResponse" />
</wsdl:message>
<wsdl:portType name="ConverterSoap">
  <wsdl:operation name="ConvertFile">
    <wsdl:input message="tns:ConvertFileSoapIn" />
    <wsdl:output message="tns:ConvertFileSoapOut" />
  </wsdl:operation>
  <wsdl:operation name="ConvertFile2">
    <wsdl:input message="tns:ConvertFile2SoapIn" />
    <wsdl:output message="tns:ConvertFile2SoapOut" />
  </wsdl:operation>
  <wsdl:operation name="ConvertFileWithData">
    <wsdl:input message="tns:ConvertFileWithDataSoapIn" />
    <wsdl:output message="tns:ConvertFileWithDataSoapOut" />
  </wsdl:operation>
  <wsdl:operation name="ConvertUrl">
    <wsdl:input message="tns:ConvertUrlSoapIn" />
    <wsdl:output message="tns:ConvertUrlSoapOut" />
  </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="ConverterSoap" type="tns:ConverterSoap">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
  <wsdl:operation name="ConvertFile">
    <soap:operation soapAction="http://pdf-tools.com/ws/o2p/ConvertFile"
                    style="document" />
    <wsdl:input>
      <soap:body use="literal" />
```

```
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="ConvertFile2">
      <soap:operation soapAction="http://pdf-tools.com/ws/o2p/ConvertFile2"
                      style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="ConvertFileWithData">
      <soap:operation soapAction="http://pdf-tools.com/ws/o2p/
                      ConvertFileWithData" style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="ConvertUrl">
      <soap:operation soapAction="http://pdf-tools.com/ws/o2p/ConvertUrl"
                      style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:binding name="ConverterSoap12" type="tns:ConverterSoap">
    <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="ConvertFile">
      <soap12:operation soapAction="http://pdf-tools.com/ws/o2p/ConvertFile"
                        style="document" />
      <wsdl:input>
        <soap12:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="ConvertFile2">
      <soap12:operation soapAction="http://pdf-tools.com/ws/o2p/ConvertFile2"
                        style="document" />
      <wsdl:input>
        <soap12:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
```

```
    <wsdl:operation name="ConvertFileWithData">
      <soap12:operation soapAction="http://pdf-tools.com/ws/o2p/
                        ConvertFileWithData" style="document" />
      <wsdl:input>
        <soap12:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="ConvertUrl">
      <soap12:operation soapAction="http://pdf-tools.com/ws/o2p/ConvertUrl"
                        style="document" />
      <wsdl:input>
        <soap12:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="Converter">
    <wsdl:port name="ConverterSoap" binding="tns:ConverterSoap">
      <soap:address location="http://dcvs/o2p/Converter.asmx" />
    </wsdl:port>
    <wsdl:port name="ConverterSoap12" binding="tns:ConverterSoap12">
      <soap12:address location="http://dcvs/o2p/Converter.asmx" />
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

5.2 Web.config

The `web.config` allows you to adjust settings according to your preferences.

> **Note:** The installer makes some adjustments to reflect the installation and log file location (as indicated by the place holder `[INSTALLDIR]`).

```
<?xml version="1.0"?>
<configuration>
    <appSettings>
        <!--
              Settings used by the web service' ConvertUrl method.
              - ResultLifeTime: the number of seconds to keep the conversion
                results available for download via the returned URL
                Default: 30 seconds.
              - TempDirectory: a path to a file system folder where the method
                shall store temporary files.
                If not specified, the default temporary folder is used.
              - ExpireAfterDownload: the number of seconds to keep the
                conversion result after the download is performed.
                If not specified, the expiration according to
                'ResultLifeTime' remains in force.
```

```
            - LogFile: full path and name of log file. Place holder
              "DATE" will be replaced by current date. Empty or missing
              value: no logging.
            - LogLevel: 1=debug, 2=info, 3=error, 4=no logging
        -->
        <add key="ResultLifeTime" value="60"/>
        <add key="TempDirectory" value="[INSTALLDIR]Temp"/>
        <add key="ExpireAfterDownload" value="0"/>
        <add key="LogFile" value="[INSTALLDIR]log\webservice-DATE.log"/>
        <add key="LogLevel" value="3"/>
    </appSettings>
    <connectionStrings/>
    <system.webServer>
        <security>
            <requestFiltering>
                <!--
                    maxAllowedContentLength limits the maximum size of the
                    request content in Bytes. 41943040 corresponds to 40 MB
                -->
                <requestLimits maxAllowedContentLength="41943040"/>
            </requestFiltering>
        </security>
    </system.webServer>
    <system.web>
        <!--
            maxRequestLength limits the maximum size for the request packet
            in kB. 40960 corresponds to 40MB.  executionTimeout sets the
            timeout for an HTTP request (600 seconds = 10 minutes)
        -->
        <httpRuntime maxRequestLength="40960" executionTimeout="600"/>
        <!--
            Set compilation debug="true" to insert debugging
            symbols into the compiled page. Because this
            affects performance, set this value to true only
            during development.
        -->
        <compilation debug="false" tempDirectory="[INSTALLDIR]Temp"/>
     <!--compilation debug="true" targetFramework="4.0"/-->
        <!--
            The <authentication> section enables configuration
            of the security authentication mode used by
            ASP.NET to identify an incoming user.
        -->
        <authentication mode="Windows"/>
        <!--
            The <customErrors> section enables configuration
            of what to do if/when an unhandled error occurs
            during the execution of a request. Specifically,
            it enables developers to configure html error pages
            to be displayed in place of a error stack trace.

        <customErrors mode="RemoteOnly"
                      defaultRedirect="GenericErrorPage.htm">
            <error statusCode="403" redirect="NoAccess.htm" />
            <error statusCode="404" redirect="FileNotFound.htm" />
        </customErrors>
        -->
        <pages controlRenderingCompatibilityVersion="3.5"
```

```
                clientIDMode="AutoID"/>
    </system.web>
</configuration>
```

6 Samples

6.1 C

This sample shows the basic call sequence and usage of the API functions. Error handling should follow each call.

```c
#include "o2pproxyapi_c.h"
#include "o2perror.h"
void CheckLastError(O2PJob job)
{
    unsigned int iError;
    const char* pszText;
    O2PJobGetLastErrorA(job, &iError, &pszErrorText);
    if (iError)
        fprintf(stderr, "Error code: 0x%x\nError text: %s\n", iError, pszText);
}

int main(int argc, char* argv[])
{
  TO2PConverter* converter = O2PCreateConverter(NULL);
  TO2PJob* job = O2PConverterCreateJob(converter);
  O2PJobSetOptionsA(job, "#;");
  O2PJobCreateOutput(job, argv[argc-1]);
  if(!O2PJobAppendDoc(job, argv[1], NULL))
  {
      CheckLastError(job);
  }
  O2PJobClose(job);
  CheckLastError(job);
  O2PJobDestroyObject(job);
  O2PDestroyConverter(converter);
  return 0;
}
```

6.2 C# .NET

```c#
using Pdftools.Converter;

try {
    job = ConverterFactory.GetInstance(
        "tcp://servername:7981/O2PService").CreateJob();
}
catch(System.Net.Sockets.SocketException se) {
    ErrText.Value = "Conversion service not available; " + se.Message;
    return;
}
ErrorInfo ei;
if (!job.SetOptions("PDFA"))
{
    ei = job.GetLastError();
    CheckError(ei); // application specific error handling
}
```

```
job.CreateOutput(null);
byte[] DocBytes = System.IO.File.ReadAllBytes("c:\\test.docx");
string DocOptions = "ORIGINALNAME=test.docx";
if (!job.AppendDoc(DocBytes, DocOptions)) {
    ei = job.GetLastError();
    CheckError(ei);
}
job.FinishConversion();
ei = job.GetLastError();
CheckError(ei);
byte[] outBytes = job.RetrieveOutput();
job.Terminate();
```

The above code sample shows an excerpt from an ASP.NET application that uses the .NET interface of the Document Converter API.

> **Note:** This is the bare remoting interface without any client side logic as in the C or COM interfaces. Retrieving error information e.g. is a remote call here, while the C and COM interfaces are caching that information.

To actually run this code from within a .NET executable or ASP.NET, the `O2PProxyNET.DLL` must either be explicitly referenced from the application, or it must be registered in the .NET Global Assembly Cache.

6.3 Visual Basic Script

```
Set oConv = CreateObject("O2PProxyAPI.ConverterProxy")
oConv.ServicePoint = "tcp://servername:7981/O2PService"
Set oJob = oConv.CreateJob()
If oJob.Create("output.pdf") Then
  oJob.AppendDoc "Document1.rtf",""
  oJob.Close
Else
  WScript.Echo "Create failed: " & oJob.ErrorText
End If
```

> **Note:** In order to execute this script, the `O2PProxyNET.DLL` must be registered in the .NET Global Assembly Cache.

6.4 Java

6.4.1 Microsoft.NET based Java API

This program sample illustrates the use of the Microsoft.NET based Java interface that is available only on Windows platforms. Java applications that are hosted on other platforms will need to make use of the web service interface.

```
import com.pdftools.converter.*;
public class DocConv
```

```
{
  public static void main(String[] args) throws Exception
  {
    if(args.length < 2) {
      System.out.println("Usage:  DocConv in1.pdf in2.pdf... out.pdf");
      return;
    }
    Converter conv = Converter.createConverter("tcp://srv:7981/O2PService");
    Job job = conv.createJob();
    job.createOutput(args[args.length-1]);
    for(int i = 0; i < args.length-1; i++)
      job.appendDoc(args[i], "");
    job.close();
    job.destroyObject();
    conv.destroyObject();
  }
}
```

Note: In order to execute this java program, `O2PProxyNET.DLL` must be registered in the .NET Global Assembly Cache (unless a copy of `O2PProcxNET.DLL` is located in the same folder as `java.exe`).

7 Examples

Examples are written in C#. The call sequence in other programming languages is identical.

Both samples require the following namespace:

```
using Pdftools.Converter;
```

7.1 Local, by reference

```
IJob job;
IConverterService converter;
try
{
    converter = ConverterFactory.GetInstance(null);
    job = converter.CreateJob();
}
catch (System.Net.Sockets.SocketException se)
{
    MessageBox.Show("Conversion service not available: " + se.Message);
    return;
}
job.SetOptions("PDFA");
job.CreateOutput(@"C:\temp\output.pdf");
if (!job.AppendDoc(@"C:\temp\input.doc", "FitToPage"))
{
    ErrorInfo err = job.GetLastError();
    MessageBox.Show("AppendDoc failed: " + err.ErrorText);
}
job.FinishConversion();
job.Terminate();
```

7.2 Remote, by value

```
IJob job;
IConverterService converter;
try
{
    converter = ConverterFactory.GetInstance("tcp://localhost:7981/
                O2PService");
    job = converter.CreateJob();
}
catch (System.Net.Sockets.SocketException se)
{
    MessageBox.Show("Conversion service not available: " + se.Message);
    return;
}
job.SetOptions("PDFA;ORIGINALNAME=" + @"C:\temp\input.doc");
job.CreateOutput(null);
```

```
// Write document to byte array
System.IO.File.ReadAllBytes inFile;
byte[] pIn;

inFile = new System.IO.File.ReadAllBytes(@"C:\temp\input.doc",
         System.IO.FileMode.Open, System.IO.FileAccess.Read);
pIn = new Byte[inFile.Length];
inFile.Read(pIn, 0, (int)inFile.Length);
inFile.Close();

// Send document (as byte array) to server
if (!job.AppendDoc(pIn, "FitToPage"))
{
    // Important: document option "ORIGINALNAME=input.doc" needs to be set
    ErrorInfo err = job.GetLastError();
    MessageBox.Show("AppendDoc failed: " + err.ErrorText);
}
job.FinishConversion();

// Retrieve converted document (as byte array)
byte[] pOut = job.RetrieveOutput();
job.Terminate();

// Write byte array to a local file
System.IO.File.ReadAllBytes outFile;
try
{
    outFile = new System.IO.File.ReadAllBytes(@"C:\temp\output.pdf",
              System.IO.FileMode.Create, System.IO.FileAccess.Write);
    outFile.Write(pOut, 0, pOut.Length);
    outFile.Close();
}
catch (System.Exception exp)
{
    MessageBox.Show("{0}", exp.Message);
}
```

7.3 Web service client (Java)

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.net.URL;
import javax.xml.namespace.QName;
import com.pdf_tools.ws.o2p.*;

/**
 * This java sample shows how to use the Document Converter web service
 * interface. The com.pdf_tools.ws.o2p package was created using the
 * wsinport tool:
 * wsimport.bat -Xendorsed -extension  -keep http://wshostname/
 * Converter.asmx?wsdl
 */
public class convert {

    public static void main(String[] args) {
```

```
        if (args.length < 2)
        {
            System.out.println("Usage:    convert [Options] path-to-input-
                                          document path-to-output-file");
            System.out.println("Options:  -j ConverterOptions");
            System.out.println("          -byname  (pass files by path name)");
            System.out.println("          -ax xmpdata.xml");
            System.out.println("          -px paramname param.data");
            System.out.println("          -sp http://<servername>/
                                              Converter.asmx?wsdl");
            return;
        }

        String input_file = null;
        String output_file = null;
        String xmp_file = null;
        String param_name = null;
        String param_file = null;
        String options = "";
        Boolean byName = false;
        String sp = "https://docws.pdf-tools.com/Converter.asmx?wsdl";

        int nr = 0;
        while (nr < args.length)
        {
            if ("-j".equals(args[nr]) && nr + 1 < args.length)
                options = args[++nr];
            else if ("-sp".equals(args[nr]) && nr + 1 < args.length)
                sp = args[++nr];
            else if ("-byname".equals(args[nr]))
                byName = true;
            else if ("-ax".equals(args[nr]) && nr + 1 < args.length)
                xmp_file = args[++nr];
            else if ("-px".equals(args[nr]) && nr + 2 < args.length)
            {
                param_name = args[++nr];
                param_file = args[++nr];
            }
            else if (input_file == null)
                input_file = args[nr];
            else if (output_file == null)
                output_file = args[nr];
            else
                System.out.println("too many parameters; skipped '" +
                args[nr] + "'");
            nr++;
        }

        try
        {
            URL url = new URL(sp);
            Converter conv = new Converter(url,
                             new QName("http://pdf-tools.com/ws/o2p/",
                                       "Converter"));
            ConverterSoap ws = conv.getConverterSoap12();
            byte[] xmpData = null;
            byte[] paramData = null;
            if (xmp_file != null)
```

```
            {
                FileInputStream xis = new FileInputStream(xmp_file);
                xmpData = new byte[xis.available()];
                xis.read(xmpData);
                xis.close();
            }
            if (param_file != null)
            {
                FileInputStream xis = new FileInputStream(param_file);
                paramData = new byte[xis.available()];
                xis.read(paramData);
                xis.close();
            }

            int pos = input_file.lastIndexOf('/');
            if (pos < 0)
                pos = input_file.lastIndexOf('\\');
            String name = input_file;
            if (pos > 0)
                name = input_file.substring(pos + 1);
            options += ";ORIGINALNAME=" + name;
            ConversionResult result;
            if (byName)
            {
                result = ws.convertFile2(input_file, options, xmpData,
                                         output_file);
            }
            else
            {
                FileInputStream is = new FileInputStream(input_file);
                byte[] inputDocBytes = new byte[is.available()];
                int n = is.read(inputDocBytes);
                if (n != inputDocBytes.length)
                    throw new Exception("Error reading bytes for file "
                                        + input_file);
                is.close();
                if (paramData != null)
                   result = ws.convertFileWithData(inputDocBytes, options,
                            xmpData, param_name, paramData);
                else
                   result = ws.convertFile(inputDocBytes, options, xmpData);
                if (result.getErrorCode() != 0)
                {
                    System.out.println("Error: " + result.getErrorCode());
                    System.out.println(result.getErrorDescription());
                }
                byte[] outputPdfBytes = result.getDocumentBytes();
                if (outputPdfBytes != null)
                {
                    FileOutputStream os = new FileOutputStream(output_file);
                    os.write(outputPdfBytes);
                    os.close();
                }
            }
            int nPages = result.getNumPages();
            System.out.print(Integer.toString(nPages));
            System.out.print(" page");
            if (nPages != 1)
```

```
                System.out.print("s");
            System.out.println(" converted");
        }
        catch(Exception e1)
        {
            System.out.println(e1);
        }
    }

}
```

8 Version history

8.1 Changes in versions 6.19–6.24

- **Update** license agreement to version 2.9

8.2 Changes in versions 6.13–6.18

No functional changes.

8.3 Changes in versions 6.1–6.12

No functional changes.

8.4 Changes in version 5

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

8.5 Changes in version 4.12

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

8.6 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** repair of corrupt image streams.

8.7 Changes in version 4.10

- **Improved** robustness against corrupt input PDF documents.
- **Improved** annotation appearance generation for polyline, squiggly, and stamp annotations.
- **Removed** the font `ZapfDingbats.ttf` from the product kit as it is not required anymore.

8.8 Changes in version 4.9

- **Improved** support for and robustness against corrupt input PDF documents.
- **Improved** repair of embedded font programs that are corrupt.
- **New** support for OpenType font collections in installed font collection.
- **Improved** metadata generation for standard PDF properties.

8.9 Changes in version 4.8

- **Changed** option TRANSFORM: Add additional parameters as command line arguments to the transform process call.
- **Changed** option `FitToPage` for Excel: Fit the width of the document or fit width and height of the document.
- **New** option `ShowComments`: Show or hide comments from Word documents in the output document.
- **New** feature: Basic support for PDF collections, the first document will be converted.
- **New** feature: Validate Office documents (format 97-2003) using the Microsoft Office Binary File Format Validator.
- **New**: Support Windows Server 2016 (RDP security).
- **Changed** option `MAILHEADER`: New value `attach`, the mail header will be embedded (PDF output) or added as text document (ZIP output).
- **Improved** creation of annotation appearances to use less memory and processing time.
- **Added** repair functionality for TrueType font programs whose glyphs are not ordered correctly.

9 Licensing, copyright, and contact

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

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

Contact

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