



3-Heights™ Document Converter API

Version 2.0

User's Manual

Contact: pdfsupport@pdf-tools.com

Owner: **PDF Tools AG**
Kasernenstrasse 1
8184 Bachenbülach
Switzerland
www.pdf-tools.com

March 29, 2011

1 Table of Content

1	Table of Content	2
2	Introduction	3
2.1	Description.....	3
2.2	Installation.....	3
3	Programmer's Reference .NET	4
3.1	Interface IConverterService.....	4
	Create.....	4
	GetKnownFileExtentsions.....	4
	GetLastError.....	4
	ProcessConversion.....	4
3.2	Class ConverterFactory.....	5
	GetInstance.....	5
3.3	Interface IJob.....	5
	AppendDoc.....	5
	Cancel.....	5
	CreateOutput.....	6
	FinishConversion.....	6
	GetLastError.....	6
	RetrieveMeta.....	6
	RetrieveOutput.....	6
	SetDocMetadata.....	7
	SetOptions.....	7
	Terminate.....	7
3.4	Struct ErrorInfo.....	7
	ErrorCode.....	7
	ErrorText.....	8
	ErrorInfo.....	8
4	Examples	9
4.1	Local, By Reference.....	9
4.2	Remote, By Value.....	10

March 29, 2011

2 Introduction

2.1 Description

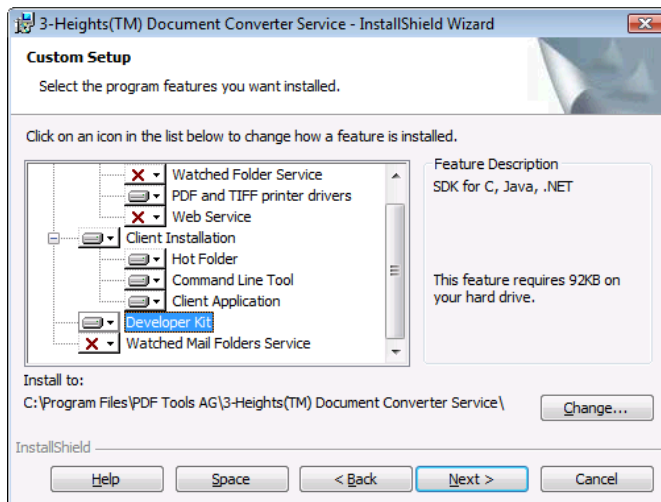
This manual provides information about how to use the API of the 3-Heights™ Document Converter Enterprise. The API is not supported by the 3-Heights™ Document Converter Desktop or SME version.

The 3-Heights™ Document Converter Enterprise manual can be found here:

<https://www.pdf-tools.com/public/downloads/manuals/dcve.pdf>

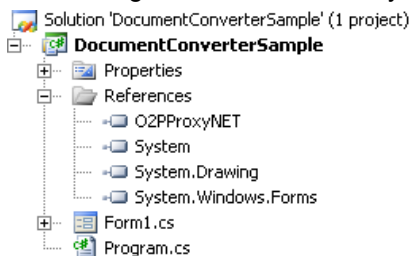
2.2 Installation

The API is installed as part of the 3-Heights™ PDF Document Converter Enterprise. During the installation, select “Custom Installation” and enable “Developer Kit”, as indicated in the screenshot below.



This installs the O2PPProxyAPI.DLL, which incorporates programming interfaces for COM and C.

When using .NET, add O2PPProxyNET.dll as a .NET reference to your project.



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

```
using Pdftools.Converter;
```

3 Programmer's Reference .NET

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](#).

3.1 Interface IConverterService

An instance of an IConverterService is created using the class [ConverterFactory](#).

Create

```
IJob CreateJob ()
```

Create a conversion job and return its interface.

GetKnownFileExtensions

```
string GetKnownFileExtensions ()
```

Retrieve the file extensions that are known to be supported by the document converter service.

Return value

A colon separated list of file extension strings

Example

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

GetLastError

```
ErrorInfo GetLastError ()
```

Get the last error code.

ProcessConversion

```
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.

Parameter

OfficeDocumentPath: The file system path of the input document

Options: The processing options

Params: Any parameters to be forwarded to plug-ins

ResultPdfPath: The file system path where the output shall be saved to

Return Value

The document title extracted from the document

March 29, 2011

3.2 Class ConverterFactory

The `GetInstance` method of the `ConverterFactory` class is used to obtain an instance of the [IConverterService](#) interface.

GetInstance

```
public static IConverterService GetInstance(string ServicePoint)
```

The `GetInstance` method 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.

3.3 Interface IJob

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

AppendDoc

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

Add a document to the job for conversion.

Parameter

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: Any options to be passed to the conversion process (e. g. open password, page format, etc.)

Params: Additional parameter data.

For a full list of supported options, please see chapter “Document Processing Options” in the documentation [dcve.pdf](#).

Return Value

True on success, false otherwise

Cancel

```
void Cancel ()
```

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

March 29, 2011

CreateOutput

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: 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.

Return Value

If a document was created successfully, this function returns true, otherwise it returns false.

Example

Create document on file system:

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

Create document in memory:

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

FinishConversion

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.

Return Value

The number of converted pages

GetLastError

ErrorInfo GetLastError()

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

Return Value

It returns an [ErrorInfo](#) structure consisting of a code and a text.

RetrieveMeta

byte[] RetrieveMeta()

Retrieve any meta information gathered during conversion processing.

Return Value

The meta information is an 8 bit ASCII buffer

RetrieveOutput

byte[] RetrieveOutput()

Retrieve the output PDF once the job is completed. Must be called after [FinishConversion](#).

March 29, 2011

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 [Example: Remote](#).

SetDocMetadata

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.

Parameter

XmpMetadata: a byte array containing the XMP Metadata to be stored in the output document. The encoding of this data shall be UTF-8.

Return Value

The SetDocMetadata returns true unless null has been passed for XmpMetadata.

Note that the Metadata is not validated at this time. If PDF/A output is produced, this Metadata is subject to PDF/A conformance checking which is performed at the end of the conversion process.

SetOptions

bool SetOptions (string Options)

Set job options.

Parameter

Options: 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. Therefore, all three option strings have the same effect. The terminating semicolon is not necessary.

For a full list of supported options, please see chapter “Setting Job Options” in the documentation [dcve.pdf](#).

Return Value

True if options are valid, false otherwise.

Example

```
job.SetOptions ("PDFA;ORIGINALNAME=input.doc");
```

Terminate

void Terminate ()

This function frees all resources and disconnects the remote connection.

3.4 Struct ErrorInfo

ErrorCode

public uint ErrorCode

The error code as uint. Error codes are described in the documentation [dcve.pdf](#), chapter “Document Conversion”.

March 29, 2011

ErrorText

```
public string ErrorText
```

A descriptive English text of the error code.

ErrorInfo

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

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

March 29, 2011

4 Examples

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

Both samples require the following namespace:

```
using Pdftools.Converter;
```

4.1 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();
```

March 29, 2011

4.2 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.FileStream inFile;
byte[] pIn;
try
{
    inFile = new System.IO.FileStream(@"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();
}
catch (System.Exception exp)
{
    MessageBox.Show("{0}", exp.Message);
    return;
}

// Send document (as byte array) to server
if (!job.AppendDoc(pIn, "FitToPage"))
{
    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.FileStream outFile;
try
{
    outFile = new System.IO.FileStream(@"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);
}
}
```