PDF to XML

Software Development Kit

INVESTINTECH.COM
PDF SOLUTIONS

Contents:

1. Introduction

The main purpose of this document is to provide you with a detailed guide on how to get started with the PDF2XML Command Line Tool and how to use Sample Files for developing an application using the PDF2XML Shared library.

PDF2XML SDK is a complete package that comes with the following components:

- ❏ PDF2XML Command Line tool
- ❏ Shared resources library
- ❏ Sample files

1.1. System Requirements

The following are the requirements for using the PDF2XML SDK tools comfortably:

- ❏ Windows 7, Windows Server 2008 or newer
- ❏ x86 Architecture CPU
- ❏ 512 MiB or more of system RAM
- ❏ At least 100 MiB of Storage Space

Besides these requirements, for SDK Dynamic Link Library Sample files, it is also recommended to have a software Development Environment such as Visual Studio, Visual Studio Code, Eclipse, Borland Delphi or something similar.

1.2. Getting more information

Investintech.com, Inc. strives to provide the best possible technical support to its current and prospective customers. If you would like personal assistance, please feel free to call us or send in an email to our Customer Service or Technical Support teams.

Customer support

Telephone:	+1 416 920 5884
Hours:	9am - 6pm EST (GMT -5:00), Monday - Friday
Email:	cs@investintech.com

Technical support

Telephone:	+1 416 920 2539
Hours:	9am - 6pm EST (GMT -5:00), Monday - Friday
Email:	techsupport@investintech.com

Fax and Mailing Address

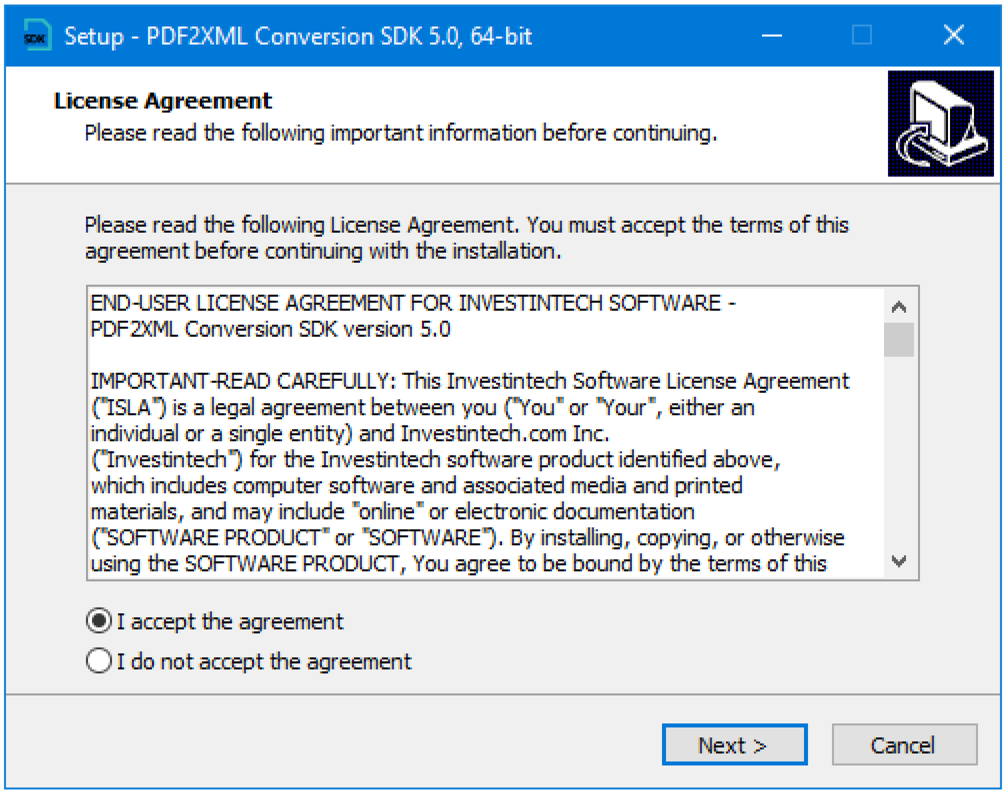
Fax:	+1 416 920 5848
Mail:	Investintech.com, Inc. 301 - 425 University Avenue Toronto, ON Canada M5G 1T6

2. Installing PDF to XML SDK tools

Thank you for choosing PDF2XML SDK tools from Investintech.com, Inc.
By following the steps below, you can quickly start using the PDF2XML SDK tool of your choice.

2.1. Windows Installer

After downloading the executable from the Investintech website, you can run the installation by starting the executable. This will run the Installation Wizard and display the Licence Agreement screen:



License Agreement
Please read the following important information before continuing.

Please read the following License Agreement. You must accept the terms of this agreement before continuing with the installation.

END-USER LICENSE AGREEMENT FOR INVESTINTECH SOFTWARE -
PDF2XML Conversion SDK version 5.0

IMPORTANT-READ CAREFULLY: This Investintech Software License Agreement ("ISLA") is a legal agreement between you ("You" or "Your", either an individual or a single entity) and Investintech.com Inc. ("Investintech") for the Investintech software product identified above, which includes computer software and associated media and printed materials, and may include "online" or electronic documentation ("SOFTWARE PRODUCT" or "SOFTWARE"). By installing, copying, or otherwise using the SOFTWARE PRODUCT, You agree to be bound by the terms of this

[✓ I accept the agreement]
I do not accept the agreement

Please read the Licence Agreement carefully. By choosing "I accept the agreement" and clicking on the Next button, you are accepting the agreement terms for usage of PDF2XML SDK.

In the next two windows, you can choose to change the location of the installation files and Start Menu folder. You are free to leave the default options.

After completing this step, you will be presented with the choice of selecting the individual components of the SDK tools you would like installed on your machine.

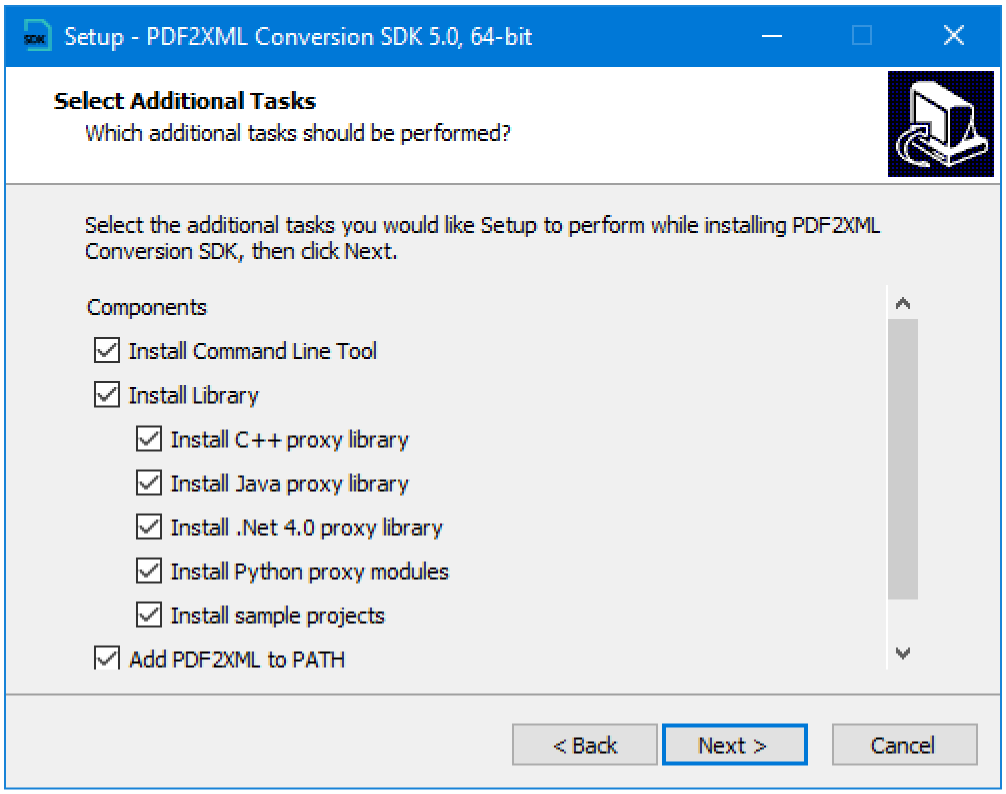
There are two major components of the SDK tools:

- ❏ Command Line Tool (CLT)
- ❏ PDF2XML Library

The Command Line tool usage is discussed in detail in Chapter 3 and the PDF2XML Library usage is discussed in Chapter 4.

By leaving the "Install Library" option checked you can also choose which of the individual proxy libraries to include with the installation. The inclusion of these libraries is strongly recommended as they significantly simplify the integration of the SDK into non native C/C++ software.

Our PDF2XML SDK tools also come with Sample Projects. These sample projects will be deployed to the "Documents" directory or more precisely, to the following location on a local machine: **`%USERPROFILE%\Documents\PDF2XML Samples`**


Setup - PDF2XML Conversion SDK 5.0, 64-bit

Select Additional Tasks

Which additional tasks should be performed?

Select the additional tasks you would like Setup to perform while installing PDF2XML Conversion SDK, then click Next.

Components

- ☑ Install Command Line Tool
- ☑ Install Library
  - ☑ Install C++ proxy library
  - ☑ Install Java proxy library
  - ☑ Install .Net 4.0 proxy library
  - ☑ Install Python proxy modules
  - ☑ Install sample projects
- ☑ Add PDF2XML to PATH

The last two steps involve a quick review of your chosen installation components and a short process of the installation itself.

After the installation finishes, simply close the installation dialog by clicking on the Finish button.
You are now ready to get started with PDF2XML SDK!

2.2. Linux and macOS Installer

In order to install PDF2XML SDK on Linux distribution or macOS, you first need to bring up your Terminal application.

In order to retrieve self extracting archive you can use wget or curl command:

```
$ wget   <executable_url>
```

or

```
$ curl <executable_url> -o pdf2xml-macos-amd64.run
```

Where `<executable_url>` is the url address of the SDK package to download.

After the .run file is downloaded you might need to adjust the permissions to allow for this file to be executed:

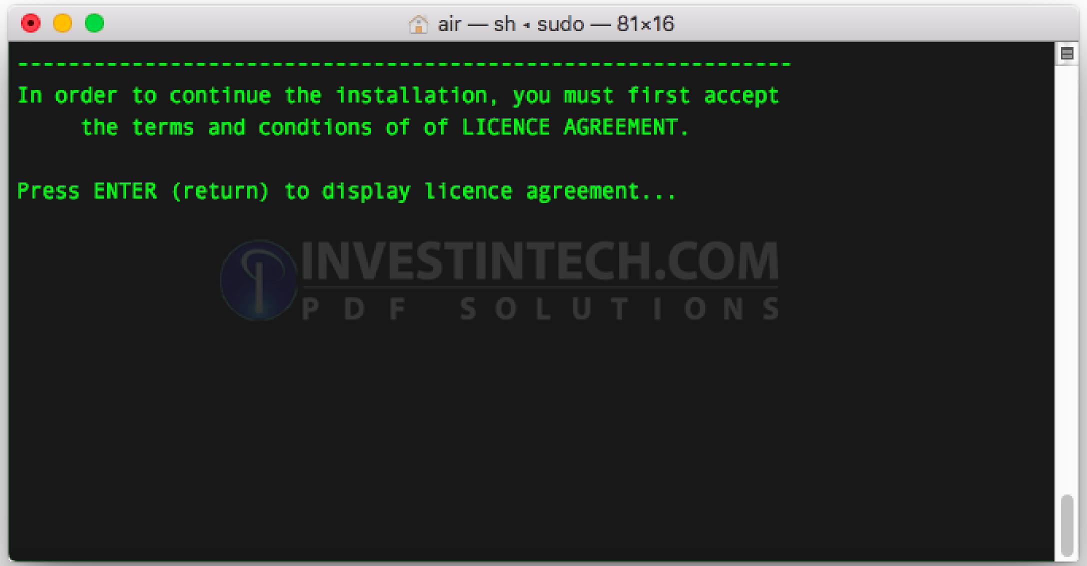
```
$ sudo chmod a+x pdf2xml-macos-amd64.run
```

Next you can extract the self extracting archive through Terminal. To start the installer simply run it as executable. For example, for macOS enter the following command in the location of the downloaded archive (or provide absolute path to the file):

```
$ sudo ./pdf2xml-macos-amd64.run
```

Same syntax is used for linux.

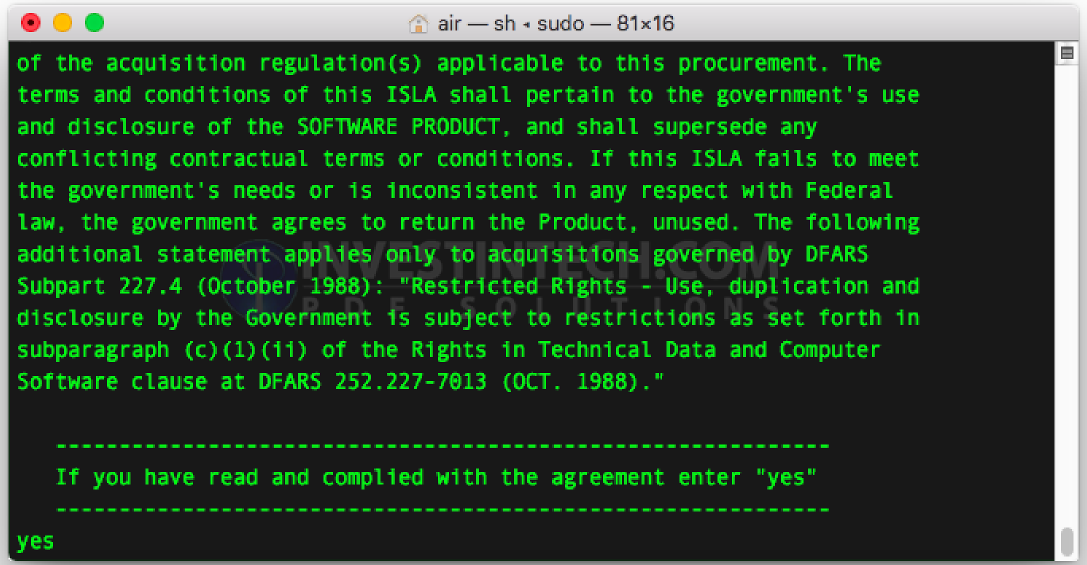
After a short extraction process, you will be prompted to preview and accept the ISLA agreement.



```
-------------------------------------------------------------
In order to continue the installation, you must first accept
     the terms and condtions of of LICENCE AGREEMENT.

Press ENTER (return) to display licence agreement...
```

Pressing enter will display the agreement. Depending on the available application for displaying the textual files on the terminal, you can navigate through the agreement using **spacebar**, **arrow down** key and/or **PgDown**.

Once you have scrolled through the agreement you will be prompted to enter "yes" in order to confirm that you agree with the terms.



```
of the acquisition regulation(s) applicable to this procurement. The
terms and conditions of this ISLA shall pertain to the government's use
and disclosure of the SOFTWARE PRODUCT, and shall supersede any
conflicting contractual terms or conditions. If this ISLA fails to meet
the government's needs or is inconsistent in any respect with Federal
law, the government agrees to return the Product, unused. The following
additional statement applies only to acquisitions governed by DFARS
Subpart 227.4 (October 1988): "Restricted Rights - Use, duplication and
disclosure by the Government is subject to restrictions as set forth in
subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer
Software clause at DFARS 252.227-7013 (OCT. 1988)."


   -------------------------------------------------------------
   If you have read and complied with the agreement enter "yes"
   -------------------------------------------------------------
yes
```

If you do not agree to the terms, you can exit the setup by entering quit, n or N.

Once you have agreed with the terms, setup will deploy the SDK Conversion file to the local file system at the location of **`/opt/`**. This directory is reported to the standard output after the successful extraction.

2.2.1. Location of binary

In order to dry run the pdf2xml binary after the extraction you can use the following command:

```
/./opt/pdf2xml/bin/pdf2xml
```

If all the dependencies are resolved, this should print out copyright information, version number and trial status, followed by "usage" (help info). For more detailed information on using Command Line Tool please refer to Section 4 and adjust to appropriate OS specific syntax.

> ***Note:*** If there is a missing resource for your installation you might have to install them manually. Common issue on Linux distribution are the font packages, which can be easily be downloaded, for example with **`sudo apt-get install libtiff5`** .

3. Registration

If you decide to purchase our PDF2XML SDK tools, you will have to perform a few more steps in order to be able to use the SDK tools without restrictions. Once you have purchased a licence through our website www.investintech.com or through one of our sales representatives, you will be provided with either a personalized Licence.lic file or a PIN number password for initializing the Libraries or both .

3.1. Using Licence file

There are two types of Licence files: the ones that require a password and the ones that do not.

If you were not provided with the information whether your licence file needs a password you can open your Licence.lic with text editor and search for </SN> tag inside. If this tag is empty you do not need a password.

Note: The licence file will contain your personalised information. Any changes done to the file will invalidate its usage, causing the tool to fall back into Trial Mode. Thus, it is recommended to keep a copy of this file safe in case of accidental file corruption.

To use the personalized License.lic file with either the Command Line Tool or Shared Libraries, it should be present in the resources directory, depending on the system:

- Under Windows copy file to the same directory as the PDF2XML.exe executable or PDF2XML shared library dll.
- Under Linux and macOS this file should be placed under ./res directory

3.2. Using Password to activate licence

For full, unhindered usage of Shared Libraries conversion methods, in addition to the Licence.lic file you need to enter the password provided when purchasing the software.

This password is provided by passing an argument to the special Initialization method which needs to be called first, before any of the conversion methods. Syntax is dependent on the type of Shared Library in question, it can be summarized to the following call:

```
Initialize (String pass);
```

4. Using the Command Line Tool

The Command line executable is a powerful, out-of-the-box tool that can be used for easy automation of conversion of PDF files into other file formats.

4.1. Preparation

In order to run the PDF2XML Command Line Tool on Windows OS, you first need to open it from the Command Prompt, Windows Powershell or some other Command Line Interface. This guide will focus on using the Command Prompt.

Open the Windows Command prompt by using any of the following techniques:

- Start typing "*command ...*" string in the Start Search bar , and clicking on the Command Prompt Desktop App when it appears.

- Start "Run" dialog using **Windows** + **R** keyboard shortcut and entering "cmd" on the keyboard and pressing "Enter" to run the cmd.exe
- Using File Explorer to navigate to the SDK installation directory and double-clicking on "Cmd.cmd" file. The installation directory is the one specified during installation.

4.1.1. Path variable (Windows)

During the installation of PDF2XML Conversion SDK you are provided with the choice for the installer to automatically set System Environment variable PATH.

If you are using Shared Libraries it is strongly recommended to have a PATH variable setup, since this makes setting up the work environment much easier. It also allows you to avoid entering an explicit path location of the PDF2XML Console Application.

If you choose not to set up a path variable during the installation you can add the location of your Shared Library and executable as a Path Environment Variable to your System Properties. PATH environment variables on Windows are accessible from the Advanced tab in the System Properties window.

4.2. Quick Single File conversion

Once you have your favorite terminal application running, converting the first file with default properties comes down to running the *PDF2XML.exe* with an absolute file path as your first parameter.

> ***Note:*** In order for the System to be able to find the correct executable make sure that Environment Variable is set up correctly (refer to the Section 4.1.1). Optionally, navigate your current location in the terminal application to the installation directory of PDF2XML SDK, or use absolute executable path.

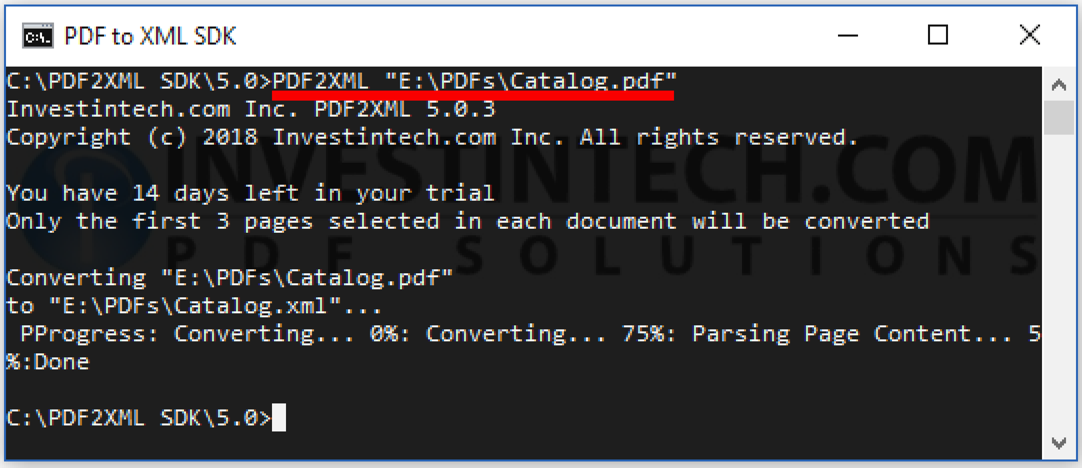
Filenames with spaces should be enclosed with double quotation marks ("") as shown in the example that follows.

Example: To convert the file named "`Catalog.pdf`" in the directory "`E:\PDFs\`" you can issue the following command:

```
>PDF2XML "E:\PDFs\Catalog.pdf"
```

In the screenshot below, you can see the expected output when the installation directory path as well as the current command line directory is: `C:\PDF2XML SDK\5.0`

During the conversion, some basic information of the process of conversion will be shown. The output will also display the location and name of the input and output files. When no additional command line arguments are provided for the output directory, the default output directory location is the same as the input directory, which in this example is "`E:\PDFs\`"



```
C:\PDF2XML SDK\5.0>PDF2XML "E:\PDFs\Catalog.pdf"
Investintech.com Inc. PDF2XML 5.0.3
Copyright (c) 2018 Investintech.com Inc. All rights reserved.

You have 14 days left in your trial
Only the first 3 pages selected in each document will be converted

Converting "E:\PDFs\Catalog.pdf"
to "E:\PDFs\Catalog.xml"...
 PProgress: Converting... 0%: Converting... 75%: Parsing Page Content... 5
%:Done

C:\PDF2XML SDK\5.0>
```

When the file conversion process is finished, "Done" will be displayed in the command line to mark the end of the conversion process. The Command Prompt will then fall back to its default state.

> ***Note:*** The output of the engine will display only the name of the output xml file, but there will be an additional resources directory containing the graphical elements of the source PDF file.

4.3. Quick Multiple File Conversion

The conversion of multiple files using PDF2XML SDK Command Line tools can be as easy as converting individual files.

The PDF2XML Console Application supports two types of wildcard characters that can be used for converting multiple files with a single command:

- An asterisk " * " character is used to match zero or more characters
- A question mark " **?** " character is used to match exactly one character

> **Note:** On Unix-like systems, the asterisk character is parsed before being passed as argument to the Terminal application. In order to avoid unpredictable behaviour, you need to either escape character * with backlash \ or enclose the argument in parentheses.

Example: Your goal is to convert all files starting with the string "Sales". For this purpose you could simply append an asterisk wildcard character to the end of a fixed string:

Sales*

Note that this would include all file types, regardless of the extension. If you would like to include only .pdf files to be converted, you could expand your string in the following way:

Sales*.pdf

Example usage of the question mark (?) wildcard character could include matching files containing exactly four characters:

????.pdf

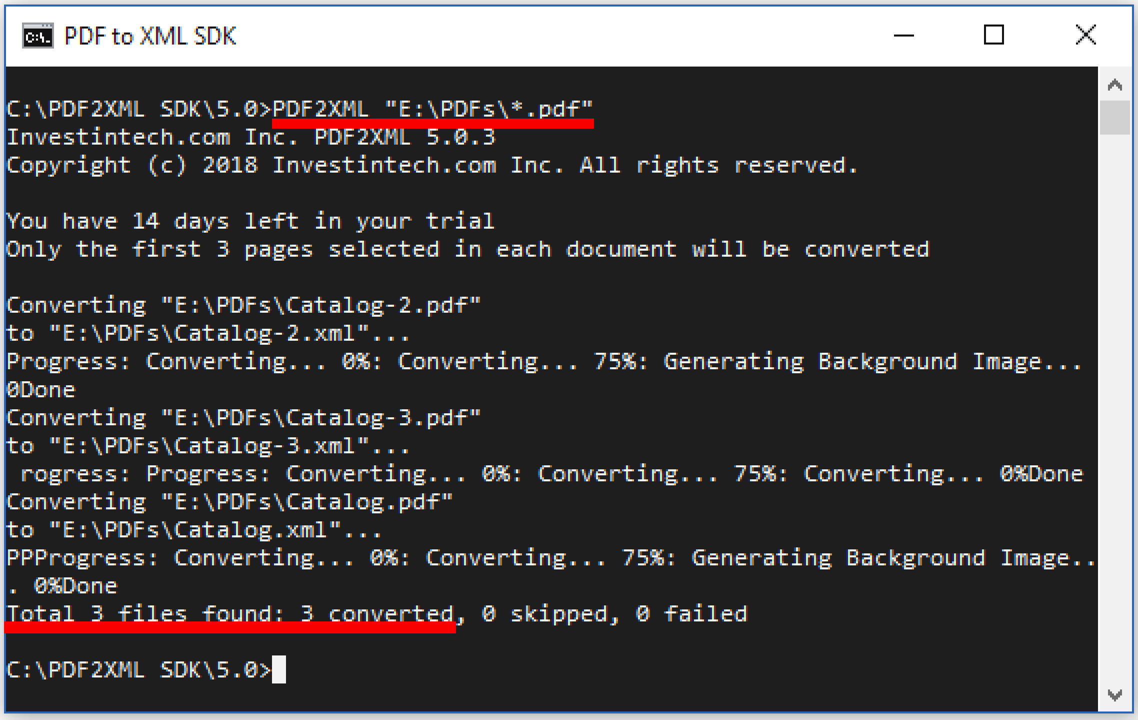
Finally, you could use the combination of both wildcards in order to match .pdf files containing three or more characters, by adapting the following syntax:

*???.pdf

Using one or more of the wildcard characters when passing on the first mandatory <inputFilePath> argument can be used to control the conversion of multiple files.

Example: To convert all files with the .pdf extension in the directory "`E:\PDFs`" you can simply issue the following command:

```
> PDF2XML "E:\PDFs\*.pdf"
```



```
C:\PDF2XML SDK\5.0>PDF2XML "E:\PDFs\*.pdf"
Investintech.com Inc. PDF2XML 5.0.3
Copyright (c) 2018 Investintech.com Inc. All rights reserved.

You have 14 days left in your trial
Only the first 3 pages selected in each document will be converted

Converting "E:\PDFs\Catalog-2.pdf"
to "E:\PDFs\Catalog-2.xml"...
Progress: Converting... 0%: Converting... 75%: Generating Background Image...
0Done
Converting "E:\PDFs\Catalog-3.pdf"
to "E:\PDFs\Catalog-3.xml"...
 rogress: Progress: Converting... 0%: Converting... 75%: Converting... 0%Done
Converting "E:\PDFs\Catalog.pdf"
to "E:\PDFs\Catalog.xml"...
PPProgress: Converting... 0%: Converting... 75%: Generating Background Image..
. 0%Done
Total 3 files found: 3 converted, 0 skipped, 0 failed

C:\PDF2XML SDK\5.0>
```

The console output is giving you information about individual file conversions, as well as a summary of the total number of converted files, skipped files and failed files (if any).

Another applicable example could be the need to convert only the pdf files that start with a "Catalog-" string, followed by exactly one other character.

```
> PDF2XML "E:\PDFs\Catalog-?.pdf"
```

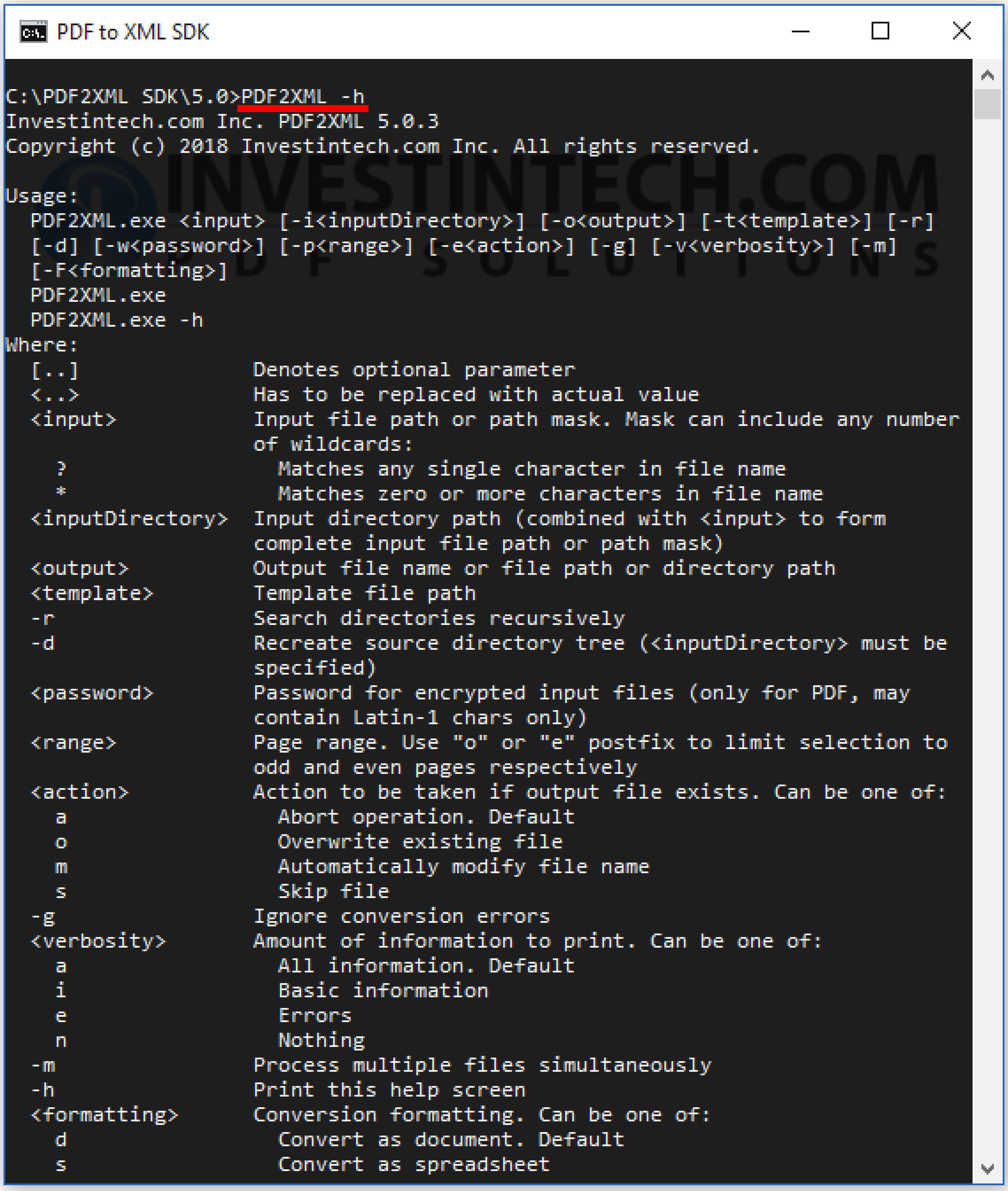
In the following chapters you will get familiar with other methods of converting multiple files using additional Command Line Arguments with the combination of wildcard characters.

4.4. Command line arguments

PDF2XML provides several command line arguments that can help with the conversion automatization fitted to the specific need of the task at hand.

To access the help menu directly from the Command Line interface simply run the PDF2XML CConsole Application without any arguments or with argument -h:

> PDF2XML



PDF to XML SDK

```
C:\PDF2XML SDK\5.0>PDF2XML -h
Investintech.com Inc. PDF2XML 5.0.3
Copyright (c) 2018 Investintech.com Inc. All rights reserved.

Usage:
  PDF2XML.exe <input> [-i<inputDirectory>] [-o<output>] [-t<template>] [-r]
  [-d] [-w<password>] [-p<range>] [-e<action>] [-g] [-v<verbosity>] [-m]
  [-F<formatting>]
  PDF2XML.exe
  PDF2XML.exe -h
Where:
  [..]              Denotes optional parameter
  <..>              Has to be replaced with actual value
  <input>           Input file path or path mask. Mask can include any number
                    of wildcards:
    ?                 Matches any single character in file name
    *                 Matches zero or more characters in file name
  <inputDirectory>  Input directory path (combined with <input> to form
                    complete input file path or path mask)
  <output>          Output file name or file path or directory path
  <template>        Template file path
  -r                Search directories recursively
  -d                Recreate source directory tree (<inputDirectory> must be
                    specified)
  <password>        Password for encrypted input files (only for PDF, may
                    contain Latin-1 chars only)
  <range>           Page range. Use "o" or "e" postfix to limit selection to
                    odd and even pages respectively
  <action>          Action to be taken if output file exists. Can be one of:
    a                 Abort operation. Default
    o                 Overwrite existing file
    m                 Automatically modify file name
    s                 Skip file
  -g                Ignore conversion errors
  <verbosity>       Amount of information to print. Can be one of:
    a                 All information. Default
    i                 Basic information
    e                 Errors
    n                 Nothing
  -m                Process multiple files simultaneously
  -h                Print this help screen
  <formatting>      Conversion formatting. Can be one of:
    d                 Convert as document. Default
    s                 Convert as spreadsheet
```

The following table contains the summary for the usage of the switches:

Switch	Parameter description	Usage
`-i`	<dirPath>	Specify the input directory
`-o`	<dirPath>	Specify the output directory
`-t`	<pcvtTemplate>	Specify the conversion template
`-r`	NONE	Recursive search of directories
`-d`	NONE	Recreate Source directory tree
`-w`	<password>	Specify the password(s) for encrypted PDF files
`-p`	<range>	Define page range
`-e`	<action>	Action to be taken if output file exists
`-g`	NONE	Ignore Conversion Errors
`-v`	<verbosity>	Amount of information to print.
`-m`	NONE	Process multiple files simultaneously
`-h`	NONE	Display help screen
`-F`	<contentFormating>	Specify content formatting

The switch parameters that are not required are marked by NONE in the above table. All other parameters are mandatory and shortly described in the above table. The function of the specific switches will be discussed in more detail in the following chapters.

> ***Note:*** When passing a parameter to a switch it must be entered immediately after the switch, that is, without spaces. It is allowed, however, to pass a parameter enclosed in double quotation marks for more readable code.

4.4.1. Specifying input directory

The Command line tool allows you to easily convert multiple files from a specified directory.
Switch **`-i`** is used to specify your source directory, after the mandatory <inputFilePath> argument.

When a directory is specified, the usage of wildcard characters is mandatory. This is required in order to match multiple files from the specified directory.

If you would like to convert all of the files in directory "`E:\PDFs`", you could use the following command:

```
>PDF2XML * -i"E:\PDFs"
```

The first argument passed is only one character. This is a simple "catch all" filter * which will match all files in the specified directory.

> ***Caution:*** When specifying a filter that is too broad, like using the catch-all asterisk sign alone, the conversion might stop when it reaches an unsupported file for conversion. In order to make sure that the conversion of the file is successful, you can do one of the following: specify the appropriate file extension or use the **-g** switch to force the engine to ignore conversion errors.

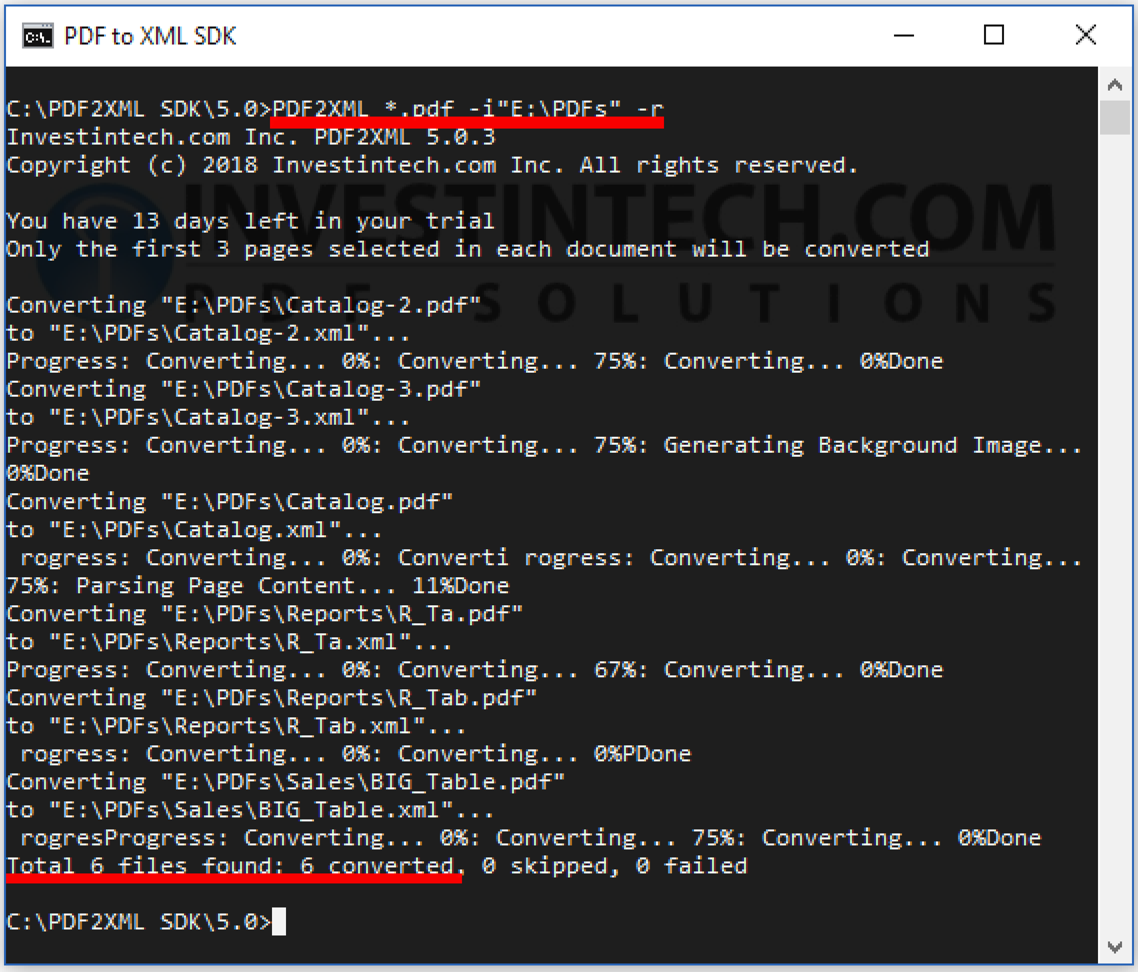
It is quite common that there are nested directories inside the specified input directory. In this case, if there are also files inside those nested directories that need to be included in the conversion output, you can use the **-r** switch to instruct the engine to perform a recursive search of files matching the pattern.

Example: If you have "`E:\PDFs`" as a directory which contains some other sub-directories, that in turn also contain files of interest, you can use the following command to easily traverse the file tree hierarchy:

```
>PDF2XML *.pdf -i"E:\PDFs" -r
```

The screenshot below shows the conversion result of the following tree hierarchy
of the "`E:\PDFs`" directory that contains two directories named "`Sales`" and "`Reports`".

Two nested folders, "`\Sales`" and "`\Reports`", also contain PDF files, that is, files that are matching the specified mask: **`"*.pdf"`**.



```
C:\PDF2XML SDK\5.0>PDF2XML *.pdf -i"E:\PDFs" -r
Investintech.com Inc. PDF2XML 5.0.3
Copyright (c) 2018 Investintech.com Inc. All rights reserved.

You have 13 days left in your trial
Only the first 3 pages selected in each document will be converted

Converting "E:\PDFs\Catalog-2.pdf"
to "E:\PDFs\Catalog-2.xml"...
Progress: Converting... 0%: Converting... 75%: Converting... 0%Done
Converting "E:\PDFs\Catalog-3.pdf"
to "E:\PDFs\Catalog-3.xml"...
Progress: Converting... 0%: Converting... 75%: Generating Background Image...
0%Done
Converting "E:\PDFs\Catalog.pdf"
to "E:\PDFs\Catalog.xml"...
 rogress: Converting... 0%: Converti rogress: Converting... 0%: Converting...
75%: Parsing Page Content... 11%Done
Converting "E:\PDFs\Reports\R_Ta.pdf"
to "E:\PDFs\Reports\R_Ta.xml"...
Progress: Converting... 0%: Converting... 67%: Converting... 0%Done
Converting "E:\PDFs\Reports\R_Tab.pdf"
to "E:\PDFs\Reports\R_Tab.xml"...
 rogress: Converting... 0%: Converting... 0%PDone
Converting "E:\PDFs\Sales\BIG_Table.pdf"
to "E:\PDFs\Sales\BIG_Table.xml"...
 rogresProgress: Converting... 0%: Converting... 75%: Converting... 0%Done
Total 6 files found: 6 converted, 0 skipped, 0 failed

C:\PDF2XML SDK\5.0>
```

The PDF2XML Command Line Tool displays the individual absolute paths for each file that is being converted.

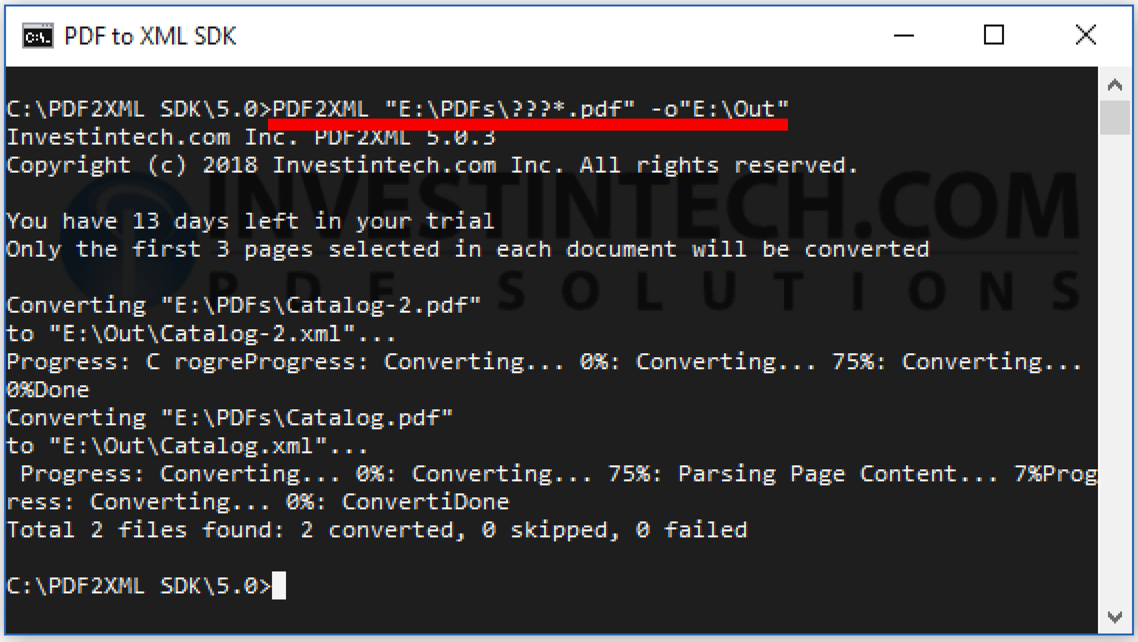
4.4.2. Specifying output directory

In addition to specifying the input folder, it is often convenient to specify the output directory to collect converted files in a different directory. By default, the converted files are placed in the same directory as the source files, which can quickly become inconvenient as the number of source files grows larger.

To save converted files in a location other than the source document directory, you can specify an absolute path of the output directory after the **-o** switch.

Example: In order to store converted files to `"E:\Out"` directory, simply declare the desired path after the **-o** switch, using similar syntax as in the following line:

```
>PDF2XML "E:\PDFs\???*.pdf" -o"E:\Out"
```


PDF to XML SDK

```
C:\PDF2XML SDK\5.0>PDF2XML "E:\PDFs\???*.pdf" -o"E:\Out"
Investintech.com Inc. PDF2XML 5.0.3
Copyright (c) 2018 Investintech.com Inc. All rights reserved.

You have 13 days left in your trial
Only the first 3 pages selected in each document will be converted

Converting "E:\PDFs\Catalog-2.pdf"
to "E:\Out\Catalog-2.xml"...
Progress: C rogreProgress: Converting... 0%: Converting... 75%: Converting...
0%Done
Converting "E:\PDFs\Catalog.pdf"
to "E:\Out\Catalog.xml"...
 Progress: Converting... 0%: Converting... 75%: Parsing Page Content... 7%Prog
ress: Converting... 0%: ConvertiDone
Total 2 files found: 2 converted, 0 skipped, 0 failed

C:\PDF2XML SDK\5.0>
```

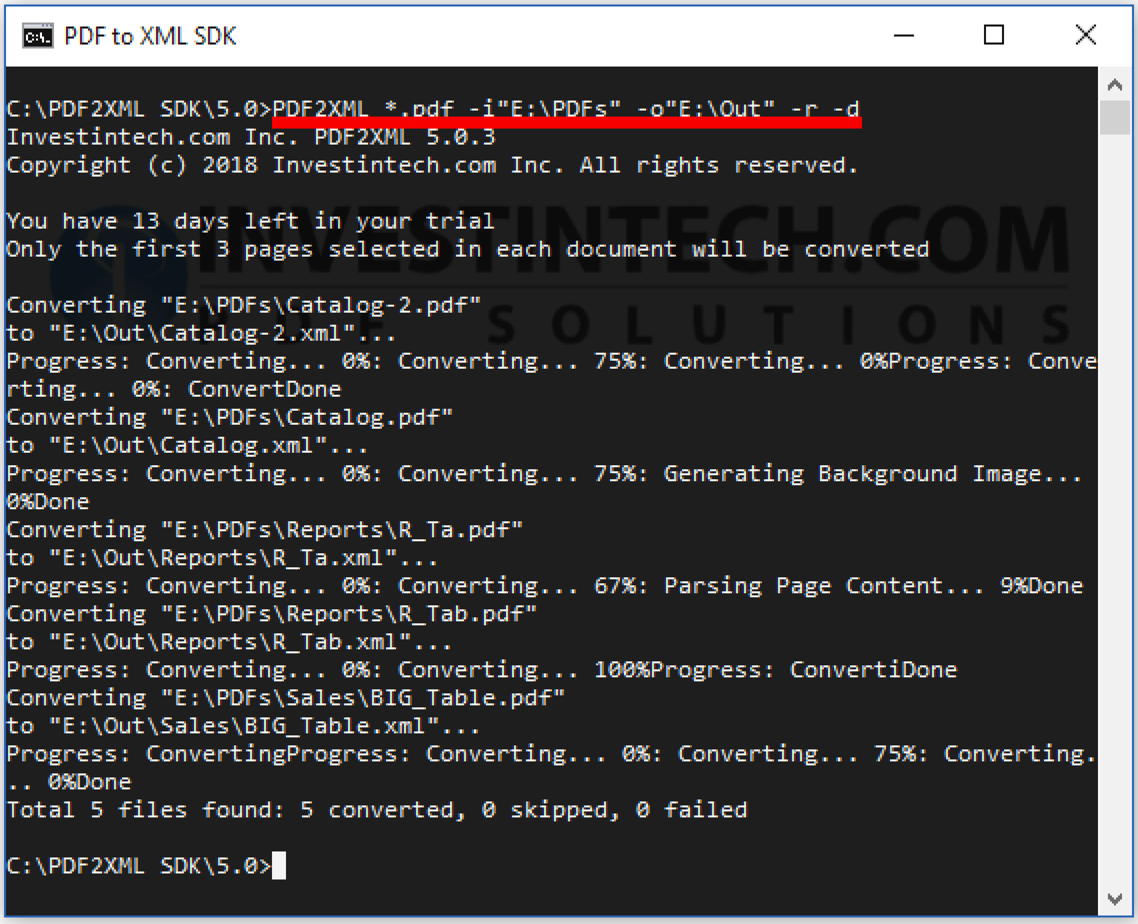
In the previous example the filter was specified for a file name using wildcards " **???*.pdf** " to match only files with the extension PDF and having at least 3 characters in the filename.

When using both <input> and <output> file paths together with the **-r** switch (Section 4.4.1) it is possible to recreate the directory tree in the output directory using the **-d** switch.

By following the simple directory tree from Section 4.4.1 where we have directory "E:\PDF Files" containing two sub-directories, "\Sales" and "\Reports", we can see how the **-d** switch can be implemented.

Example: In the following example you can see how the **-d** switch is producing a conversion of all files from all three directories, while re-creating the directory tree in the output directory:

```
>PDF2XML *.pdf -i"E:\PDFs" -o"E:\Out" -r -d
```



```
C:\PDF2XML SDK\5.0>PDF2XML *.pdf -i"E:\PDFs" -o"E:\Out" -r -d
Investintech.com Inc. PDF2XML 5.0.3
Copyright (c) 2018 Investintech.com Inc. All rights reserved.

You have 13 days left in your trial
Only the first 3 pages selected in each document will be converted

Converting "E:\PDFs\Catalog-2.pdf"
to "E:\Out\Catalog-2.xml"...
Progress: Converting... 0%: Converting... 75%: Converting... 0%Progress: Conve
rting... 0%: ConvertDone
Converting "E:\PDFs\Catalog.pdf"
to "E:\Out\Catalog.xml"...
Progress: Converting... 0%: Converting... 75%: Generating Background Image...
0%Done
Converting "E:\PDFs\Reports\R_Ta.pdf"
to "E:\Out\Reports\R_Ta.xml"...
Progress: Converting... 0%: Converting... 67%: Parsing Page Content... 9%Done
Converting "E:\PDFs\Reports\R_Tab.pdf"
to "E:\Out\Reports\R_Tab.xml"...
Progress: Converting... 0%: Converting... 100%Progress: ConvertiDone
Converting "E:\PDFs\Sales\BIG_Table.pdf"
to "E:\Out\Sales\BIG_Table.xml"...
Progress: ConvertingProgress: Converting... 0%: Converting... 75%: Converting.
.. 0%Done
Total 5 files found: 5 converted, 0 skipped, 0 failed

C:\PDF2XML SDK\5.0>
```

Note: When using this **-d** switch to recreate a directory tree, only child-directories that contain files with names that match the filter specified in the first <inputFilePath> argument will be created. This is because the **-r** switch will ignore directories with no matching file mask.

Following the previous example, if directory `"E:\PDFs"` contained a folder named `"Images"` which contains only .png files, the `"Images"` folder would not be created in the `"E:\Out"` directory, since the provided pattern, *.pdf, only matches pdf files.

4.4.3. Formatting

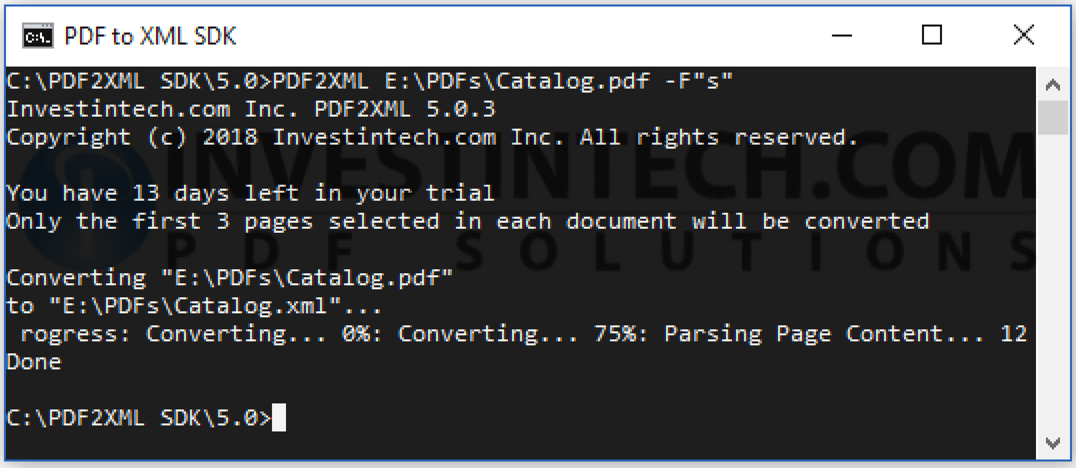
To change the formatting of your XML file you can set the **-F** flag (capital letter F) followed by the parameter. By default PDF2XML is converting files as documents, and includes resources in the separate directory.

If you prefer to have an XML document converted as a spreadsheet you can pass parameter to the **F** flag .

Example: To convert the file named "Catalog.pdf" to XML file format as spreadsheet you can use the following command:

```
>PDF2XML E:\PDFs\Catalog.pdf -F"s"
```

As shown in the screenshot below:



```
C:\PDF2XML SDK\5.0>PDF2XML E:\PDFs\Catalog.pdf -F"s"
Investintech.com Inc. PDF2XML 5.0.3
Copyright (c) 2018 Investintech.com Inc. All rights reserved.

You have 13 days left in your trial
Only the first 3 pages selected in each document will be converted

Converting "E:\PDFs\Catalog.pdf"
to "E:\PDFs\Catalog.xml"...
 rogress: Converting... 0%: Converting... 75%: Parsing Page Content... 12
Done

C:\PDF2XML SDK\5.0>
```

4.4.4. Page ranges

When dealing with large PDF files there is often no need to convert all of the pages in the document. Using Page Range switch, you can define which part of the document to convert.

With the Page Range switch -p, you can define a single page or a subset of pages to be converted. When defining a subset of pages, you can specify a start page number (<startPageNº>), end page number (<endPageNº>) or both. When specifying only the starting page, followed by a hyphen (-), the assumed end page is the end of the document. Similarly, when specifying only the last page number, the assumed starting page is the first page of the document.

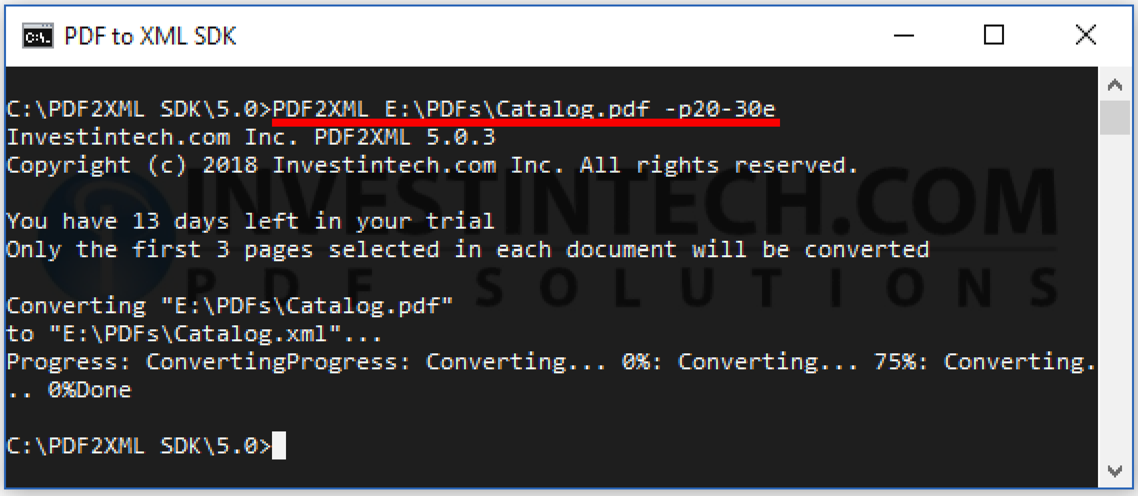
Additionally, you can decide to extract only odd or even pages of the document by using suffixes, **o** or **e** respectively. The table below provides you with an overview of the possible syntax for each case and what it does:

Syntax	**What does it do?**
`-p<SinglePageN°>`	Convert one page of the document.
`-p<startPageN°>-<endPageN°>`	Convert all pages starting with `<startPageN°>` and ending with `<endPageN°>`
`-p<startPageN°>-<endPageN°>e`	Convert even pages starting with `<startPageN°>` and ending with `<endPageN°>`
`-p<startPageN°>-<endPageN°>o`	Convert odd pages starting with `<starPageN°>` and ending with `<endPageN°>`

Note: Odd and even pages are "absolute", meaning that they are dependent on the real document page numbering, and not on `<starPageN°>`.

Example: Let's say you have a large PDF file, but you only require even pages from page 20 to page 30. You can use the following command to extract specific pages:

```
>PDF2XML E:\PDFs\Catalog.pdf -p20-30e
```



PDF to XML SDK

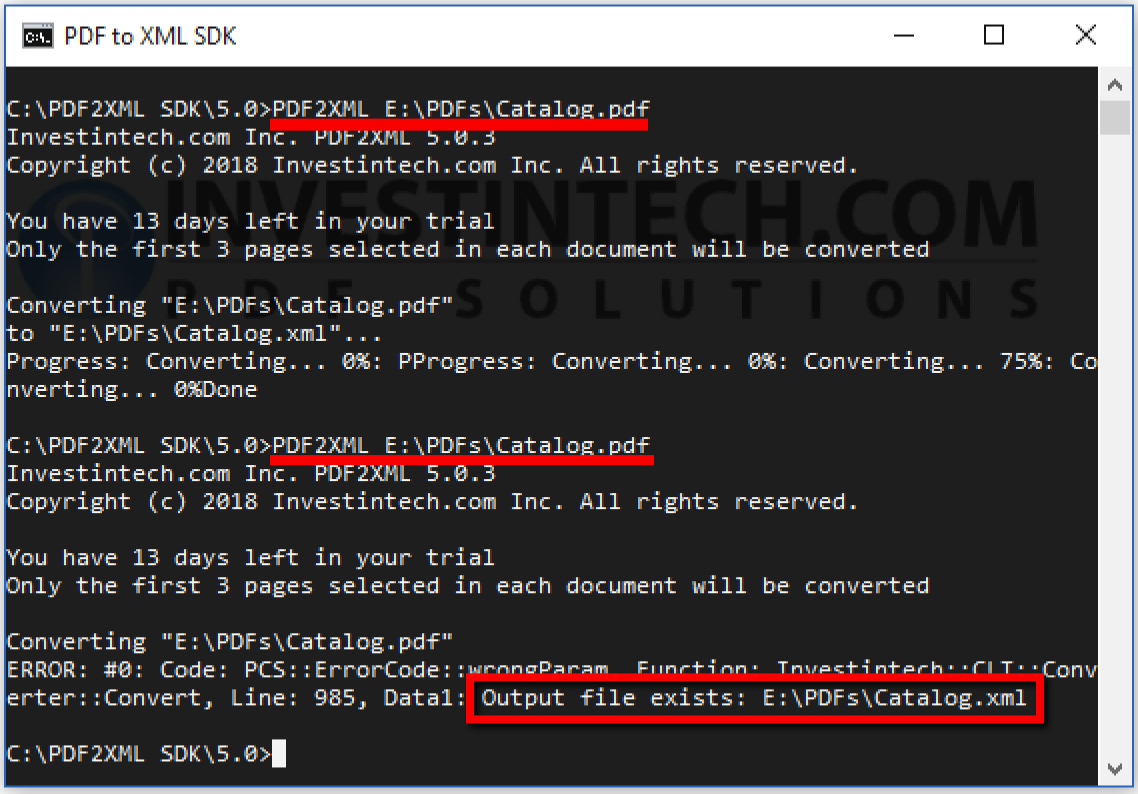
```
C:\PDF2XML SDK\5.0>PDF2XML E:\PDFs\Catalog.pdf -p20-30e
Investintech.com Inc. PDF2XML 5.0.3
Copyright (c) 2018 Investintech.com Inc. All rights reserved.

You have 13 days left in your trial
Only the first 3 pages selected in each document will be converted

Converting "E:\PDFs\Catalog.pdf"
to "E:\PDFs\Catalog.xml"...
Progress: ConvertingProgress: Converting... 0%: Converting... 75%: Converting.
.. 0%Done

C:\PDF2XML SDK\5.0>
```

4.4.5. File name collision options

When converting files with generic names, or when repeating the conversion of the same file, it is a common occurrence that the output directory already contains a file with exactly the same name.

Example: When converting the same file two times, let it be a file with absolute path `E:\PDFs\Catalog.pdf`, you will see the following error in the console:

`Output file exists: E:\PDFs\Catalog.pdf`


PDF to XML SDK

```
C:\PDF2XML SDK\5.0>PDF2XML E:\PDFs\Catalog.pdf
Investintech.com Inc. PDF2XML 5.0.3
Copyright (c) 2018 Investintech.com Inc. All rights reserved.

You have 13 days left in your trial
Only the first 3 pages selected in each document will be converted

Converting "E:\PDFs\Catalog.pdf"
to "E:\PDFs\Catalog.xml"...
Progress: Converting... 0%: PProgress: Converting... 0%: Converting... 75%: Co
nverting... 0%Done

C:\PDF2XML SDK\5.0>PDF2XML E:\PDFs\Catalog.pdf
Investintech.com Inc. PDF2XML 5.0.3
Copyright (c) 2018 Investintech.com Inc. All rights reserved.

You have 13 days left in your trial
Only the first 3 pages selected in each document will be converted

Converting "E:\PDFs\Catalog.pdf"
ERROR: #0: Code: PCS::ErrorCode::wrongParam  Function: Investintech::CLI::Conv
erter::Convert, Line: 985, Data1: Output file exists: E:\PDFs\Catalog.xml

C:\PDF2XML SDK\5.0>
```

For safety reasons, conversion of a file will fail when file name collision occurs. Using the <action> switch **-e** you can specify a different action when file collision occurs. The following table gives you a summary of the possible options:

-e Parameter	What happens in case of collision ?	Print to terminal when collide?
a	Operation will be aborted (default)	Yes
o	Overwrite existing file	Yes
m	Modify the file name to avoid collision	No
s	Skip the conversion of this file	Yes

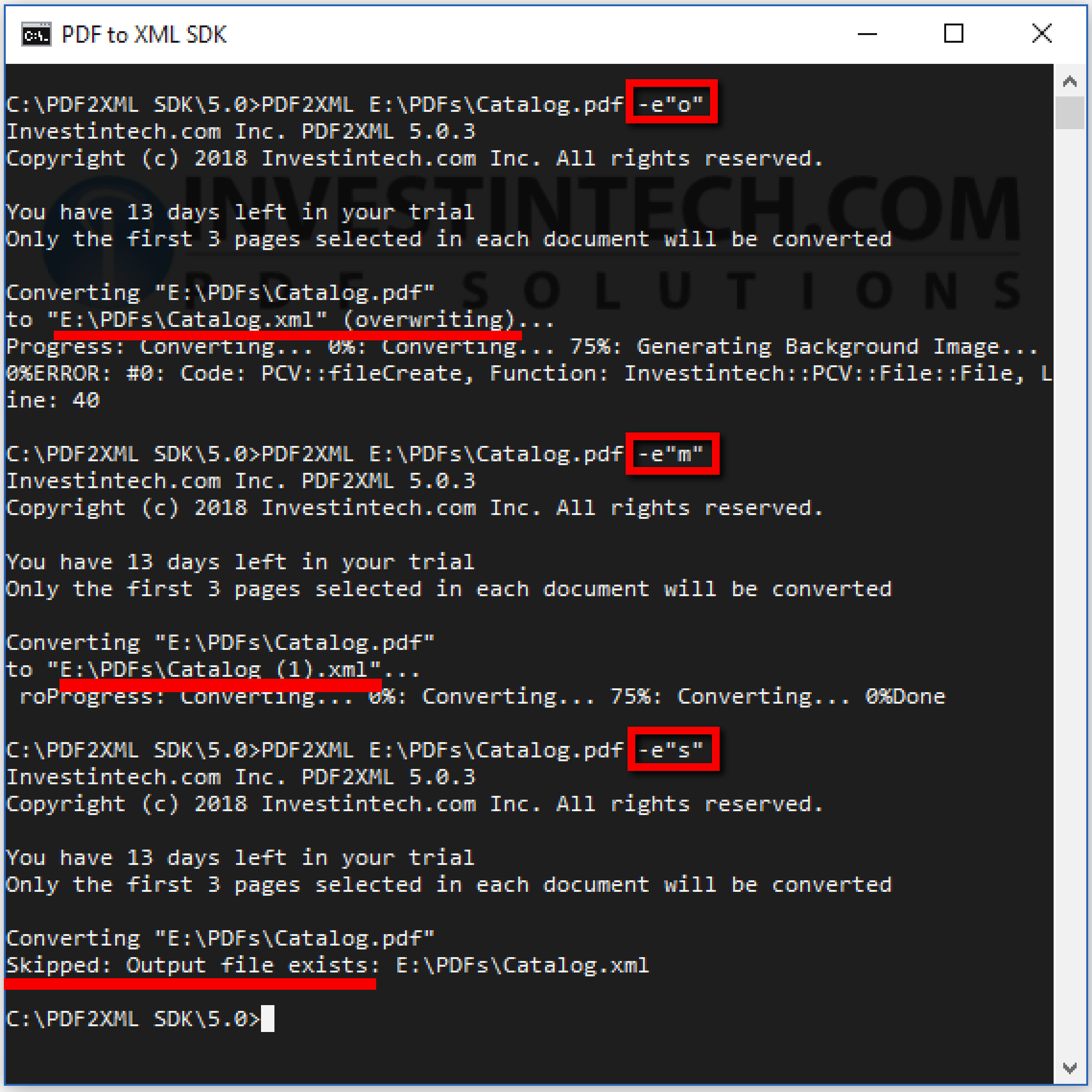
In the case of a successful file overwrite, or file skipping, the Command Line Tool will print to the console that these actions occurred, unless the verbosity level is changed explicitly (Section 4.4.6).

If a collision occurs and <action> (switch **-e**) is set to modify (parameter **m**), no special output will be displayed. However, since the name of the output file is displayed for every conversion with default verbosity level, the output file name will differ from the input file name, which points out that this specific event has occurred.

> ***Caution:*** At first glance it might appear that the default option (abort) and using the **-e**"**s**" action to skip the file when collision occurs have the same effect, with only the output to console being different. When converting a single file this will, indeed, be true. However, this is not the case when converting multiple files. If any of the files has a name collision, the rest of the files will not be converted. The exception to this rule would be when a collision occurs for the last file in the queue, in which case only that last file will not be converted.

Example: Below you can see a screenshot of the output when file collision occurs while running all three parameters to the <action> **-e** switch in sequence one by one:

```
>PDF2XML E:\PDFs\Catalog.pdf -e"o"

>PDF2XML E:\PDFs\Catalog.pdf -e"m"

>PDF2XML E:\PDFs\Catalog.pdf -e"s"
```



```
C:\PDF2XML SDK\5.0>PDF2XML E:\PDFs\Catalog.pdf -e"o"
Investintech.com Inc. PDF2XML 5.0.3
Copyright (c) 2018 Investintech.com Inc. All rights reserved.

You have 13 days left in your trial
Only the first 3 pages selected in each document will be converted

Converting "E:\PDFs\Catalog.pdf"
to "E:\PDFs\Catalog.xml" (overwriting)...
Progress: Converting... 0%: Converting... 75%: Generating Background Image...
0%ERROR: #0: Code: PCV::fileCreate, Function: Investintech::PCV::File::File, L
ine: 40

C:\PDF2XML SDK\5.0>PDF2XML E:\PDFs\Catalog.pdf -e"m"
Investintech.com Inc. PDF2XML 5.0.3
Copyright (c) 2018 Investintech.com Inc. All rights reserved.

You have 13 days left in your trial
Only the first 3 pages selected in each document will be converted

Converting "E:\PDFs\Catalog.pdf"
to "E:\PDFs\Catalog (1).xml"...
 roProgress: Converting... 0%: Converting... 75%: Converting... 0%Done

C:\PDF2XML SDK\5.0>PDF2XML E:\PDFs\Catalog.pdf -e"s"
Investintech.com Inc. PDF2XML 5.0.3
Copyright (c) 2018 Investintech.com Inc. All rights reserved.

You have 13 days left in your trial
Only the first 3 pages selected in each document will be converted

Converting "E:\PDFs\Catalog.pdf"
Skipped: Output file exists: E:\PDFs\Catalog.xml

C:\PDF2XML SDK\5.0>
```

Caution: Since conversion to XML files with default formatting (section 3.4.3) will create additional resources directory to contain images. When using parameter ***o*** to overwrite an existing file, even though the xml file itself will indeed be replaced, collision with the directory will still occur.

If collision occurs and <action> (switch **-e**) is set to modify (parameter **m**), no special output will be displayed. However, since the name of the output file is displayed for every conversion with default verbosity level, the output file name will differ from the input file name, which points out that this specific event has occurred.

Note: At first glance it might appear that the default option (abort) and using **-e**"**s**" action to skip the file when collision occurs are having the same effect, with only output to console being different. When converting a single file this will, indeed, be true. However this is not the case when converting multiple files. If any of the files has a name collision, the rest of the files will not be converted. Exception to this rule would be when collision occurs for the last file in the queue, in which case only that last file will not be converted.

4.4.6. Verbosity level of Console output

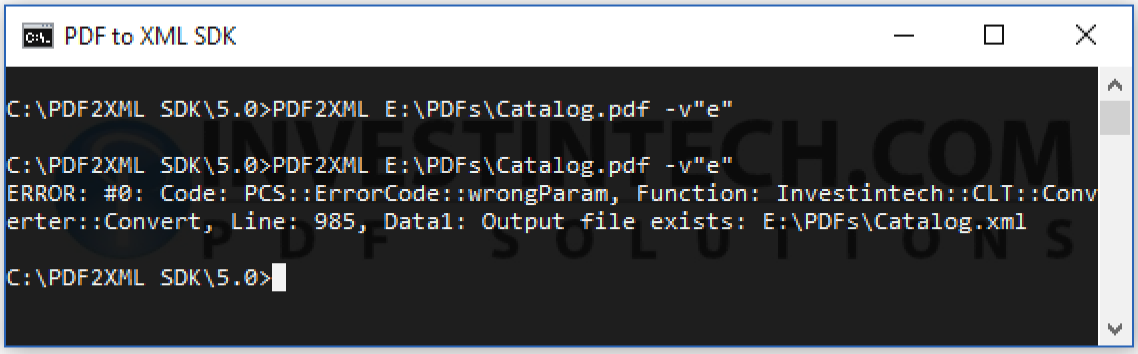
Using the Command Line Tool is often done in a fully automated manner. When this is the case, it is often unnecessary or even counterproductive to have all of the conversion outputs to be printed to the command line interface.

You can control the amount of output using the **-v** switch followed by one of the options. This is described in the table below ordered by the amount of output that will be displayed:

-v Switch parameter	**What shall be printed out?**
a	All of the information (default)
i	Only basic information
e	Only errors, if they occur
n	Nothing

Example: We can reproduce an example of the file collision from Section 4.4.5, converting the same file two times, but this time with the verbosity level reduced only to errors. The output of repeated conversion of the file "`Catalog.pdf`" is shown below:

```
>PDF2XML E:\PDFs\Catalog.pdf -v"e"
```


PDF to XML SDK

```
C:\PDF2XML SDK\5.0>PDF2XML E:\PDFs\Catalog.pdf -v"e"

C:\PDF2XML SDK\5.0>PDF2XML E:\PDFs\Catalog.pdf -v"e"
ERROR: #0: Code: PCS::ErrorCode::wrongParam, Function: Investintech::CLT::Conv
erter::Convert, Line: 985, Data1: Output file exists: E:\PDFs\Catalog.xml

C:\PDF2XML SDK\5.0>
```

You can see from the screenshot that the first run of the command didn't produce any output at all, since no error occurred, while for the second conversion we have an error, and it is the only output displayed.

> ***Important:*** Verbosity mode can have a very big impact on the performance of the Command Line Tool. Conversion time can be doubled when complete verbosity level is set to all, which is the default. When the tool is invoked by another app or script it is strongly recommended to turn off the verbosity level completely. Otherwise, you should generally use the scarcest verbosity level you require in order to get the fastest conversions.

4.4.7. Converting password protected files

PDF files are sometimes encrypted, and thus cannot be opened without providing a password. In order to be able to convert these files you can provide the password as a parameter to the **-w** switch.

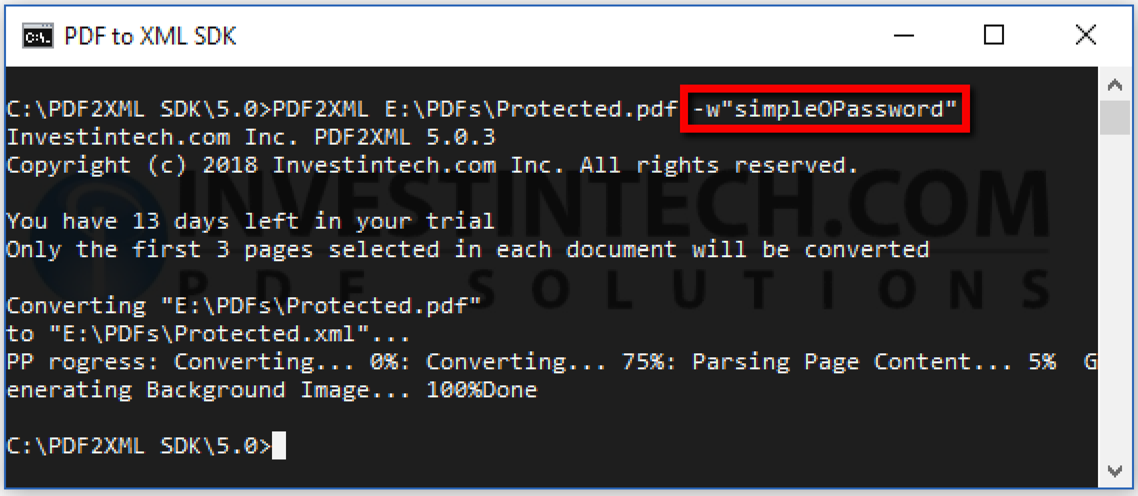
If you try to convert an encrypted file, you will get the following error:

```
ERROR: #0: Code: PDL::ERR_PDF_WRONG_PASSWORD
```

Since the conversion engine only requires reading of the document in order to convert it to another file format, you can supply either an "Owner" or "User" password to successfully convert the file.

Example: Here is a simple example of how to convert an encrypted file with an absolute file path "`E:\PDFs\Protected.pdf`", and password "`simple0Password`":

```
>PDF2XML E:\PDFs\Protected.pdf -w"simpleOPassword"
```



```
C:\PDF2XML SDK\5.0>PDF2XML E:\PDFs\Protected.pdf -w"simpleOPassword"
Investintech.com Inc. PDF2XML 5.0.3
Copyright (c) 2018 Investintech.com Inc. All rights reserved.

You have 13 days left in your trial
Only the first 3 pages selected in each document will be converted

Converting "E:\PDFs\Protected.pdf"
to "E:\PDFs\Protected.xml"...
PP rogress: Converting... 0%: Converting... 75%: Parsing Page Content... 5%  G
enerating Background Image... 100%Done

C:\PDF2XML SDK\5.0>
```

4.4.8. Ignoring Errors

Having an error reporting feature is an important part of every application. There are some cases when the conversion process needs to proceed despite the errors, especially when large scale file conversions are in question.

The correct option to use in this case is the **-g** switch that instructs the PDF2XML Command Line Tool to ignore conversion errors and continue the execution of the program.

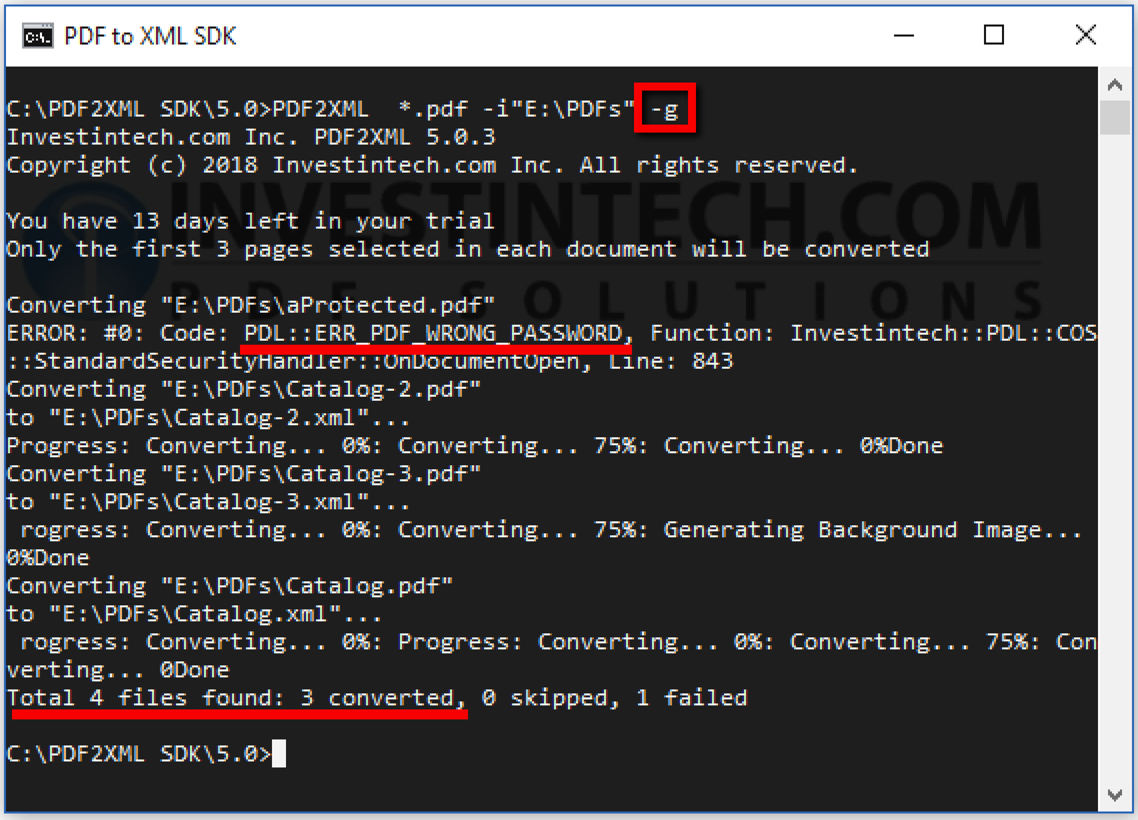
When converting multiple files using wildcard characters, it is possible that one of the files cannot be converted. When this happens, the process of conversion will stop and all the other files will not be converted.

Example: You have a directory named `"E:\PDFs\"` with several .pdf files, one of which is password protected. You can order the conversion of all .pdf files in the directory with the following command:

```
>PDF2XML  *.pdf -i"E:\PDFs"
```

If conversion fails when it reaches the password protected file, then all the other files will not be converted. To avoid this from happening you can add the **-g** switch to the command:

```
>PDF2XML  *.pdf -i"E:\PDFs" -g
```


PDF to XML SDK

```
C:\PDF2XML SDK\5.0>PDF2XML  *.pdf -i"E:\PDFs" -g
Investintech.com Inc. PDF2XML 5.0.3
Copyright (c) 2018 Investintech.com Inc. All rights reserved.

You have 13 days left in your trial
Only the first 3 pages selected in each document will be converted

Converting "E:\PDFs\aProtected.pdf"
ERROR: #0: Code: PDL::ERR_PDF_WRONG_PASSWORD, Function: Investintech::PDL::COS
::StandardSecurityHandler::OnDocumentOpen, Line: 843
Converting "E:\PDFs\Catalog-2.pdf"
to "E:\PDFs\Catalog-2.xml"...
Progress: Converting... 0%: Converting... 75%: Converting... 0%Done
Converting "E:\PDFs\Catalog-3.pdf"
to "E:\PDFs\Catalog-3.xml"...
 rogress: Converting... 0%: Converting... 75%: Generating Background Image...
0%Done
Converting "E:\PDFs\Catalog.pdf"
to "E:\PDFs\Catalog.xml"...
 rogress: Converting... 0%: Progress: Converting... 0%: Converting... 75%: Con
verting... 0Done
Total 4 files found: 3 converted, 0 skipped, 1 failed

C:\PDF2XML SDK\5.0>
```

From the output in the console, you can see that Error has occurred when a password protected file was reached. Nevertheless, the conversion of all other files was completed successfully.

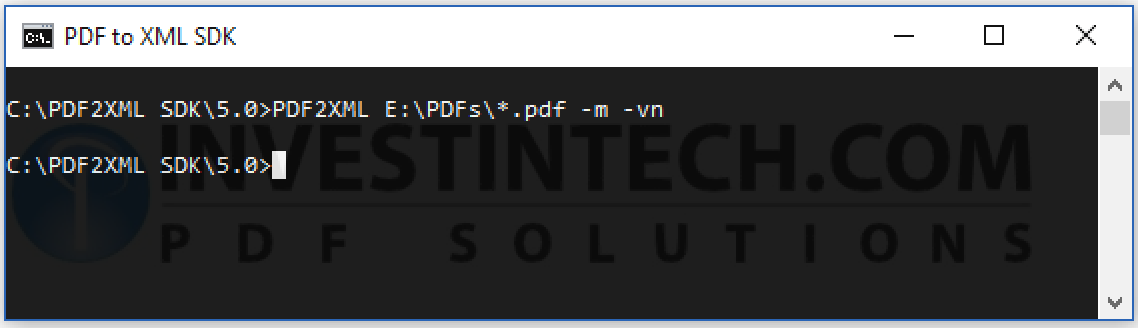
4.4.9. Parallel Conversion

One more option that can help you in the process of converting a large number of files is using the **-m** switch which forces the engine to convert multiple files simultaneously.

Since the immediate output of a big number of files can be overwhelming, this command is intended to be used together with the verbosity level (section 3.4.7) set to "none" using **-vn** (or **-v**"**n**").

To simultaneously convert all of the PDF files in a directory named `"E:\PDFs"`, use the following command:

```
PDF2XML E:\PDFs\*.pdf -m -vn
```



```
C:\PDF2XML SDK\5.0>PDF2XML E:\PDFs\*.pdf -m -vn

C:\PDF2XML SDK\5.0>
```

Of course, due to the verbosity level being set to "none", there will be no output on the command line interface.

5. Using the Shared Library

The PDF2XML SDK comes with a complete set of files which allow for quick integration into your development environment. PDF2XML.dll is a collection of methods compiled, linked and stored in a Dynamic Link Library. You can use the Dynamic Link Library PDF2XML.dll directly, or through one of the proxy libraries for COM and .NET for non C/C++ projects.

5.1. Preparation

After installing the PDF2XML SDK tool on your local machine you should have access to Sample Files in addition to libraries and executive binaries. These files can be accessed in your local "Documents" directory, the install directory relative to the User folder on your local machine, or by using the out-of the box environment variable:

```
%USERPROFILE%\Documents\PDF2XML Samples
```

The stated folder above contains subfolders with different Visual Studio sample projects.

> ***Note:*** While instructions for using sample projects can be implemented using the default location in which they are stored, it is recommended to make a copy of this sample project to some other location so you can easily rollback to a default state.

5.1.1. Licence Registration

In order to use a fully licenced version of PDF2XML.[dll | so | dylib] in your application, you should receive one or two components:

1. Personalised Licence.lic file
1. (optional)Password corresponding to the Licence file

To use the licenced application, the `Licence.lic` file needs to be present in the same directory as your executables. If you have a licence file that requires password you must pass an argument to the initialization function :

```
PDF2XML.Initialize( String pass)
```

If you do not have a Licence.lic file, you do not have to provide a password String, however, this will give you limited trial access to our SDK tools. In order to use the unlicensed SDKs, you can pass an empty string to the initialization function.

5.2. C++ sample project

This section will explain how to connect the C++ sample project with the PDF2XML.dll library. On Windows sample project is placed by default in the following directory:

```
%USERPROFILE%\Documents\PDF2XML Samples\C++
```

For Linux and macOS Sample projects are in the same directory as installation directory:

```
/opt/pdf2xml/samples
```

In order to ensure rollback it is advisable to make a copy of sample files before starting the integration.

Sample projects can be opened by double-clicking on `Sample.sln` or by opening the Visual Studio by and choosing `File -> Open -> Open/Project Solution...` and then navigating to the location of the sample project.

5.2.1. Resolving Dependencies

If dependencies are not resolved immediately additional setting is required in order to run the Sample project. For the Sample file to function it needs to know a location to `<PDF2XML.hpp>` and shared library `PDF2XML<.dll|.so|.dylib>`.

In order to ensure that Sample projects can access required resources, you might need to add them as "Additional include directories".

When using Visual Studio you can achieve this by right-clicking on the Solution project in the Solution Explorer (right hand side by default). From the drop down menu choose "Properties."

On the left side choose `C/C++ -> General.` To edit the first item click on down arrow in row Additional Include Directories and choose `<Edit>`

In the new window make sure that at least two directories are present. One is the location of the PDF2XML.dll and the other is the location of the header file `<PDF2XML.hpp>`. Both files are deployed during the installation.

Example: If your installation directory during the setup was `C:\PDF2XML SDK\5.0\`, then following two entries should be present in the **Additional Include Directory** section:

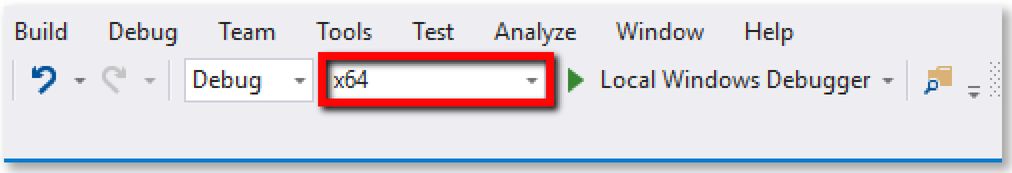
```
C:\PDF2XML SDK\5.0\

C:\PDF2XML SDK\5.0\Sources\C++
```

After build dependencies are resolved, linker might need a path to the directory containing the `PDF2XML.lib` file. In order to provide an explicit path to the linker you can bring up Properties of the Sample project and goto `Linker->General` and under **Additional Library Directories** make sure to add the root installation directory of the PDF2XML SDK.

Example: If your installation directory during the setup was `C:\PDF2XML SDK\5.0\`, then following entry should be present in the **Additional Library Directories** section:

```
C:\PDF2XML SDK\5.0\
```

When dependencies are resolved you can use the sample application.



When compiling the Sample file make sure to use the configuration that matches the target platform of the PDF2XML SDK tools.

5.2.2. Running Sample files (Visual Studio)

In order to run a Sample application out of the box at least two arguments containing a path to the PDF file and path to the converted output file.

You can provide that file manually to the compiled executable through console application or through Visual Studio debugger. Optionally, you can also hard-code the path directly into the sample file.

1. Directly run the `...\Debug\Sample.exe` executable from Command Line, which Visual Studio built for you, and pass the required parameter.

2. Open the Sample Project Property page by right clicking on Sample project from the Solution Explorer window and choosing Properties from the context menu. (You must select Project properties, not Solution properties).
From the Sample Project Property dialog, on the left hand side, choose "Debugging" and fill in the path for the Command Arguments field on the right. The first Argument is the path to the Input File that should be loaded.

Once you have provided a valid file path, you can start to build and run the Sample application.
Sample application will open the file, output some of the meta information of the PDF file to the console, edit the file in several ways, as well as render the first page of the document. Rendered file is saved in jpeg format in the same directory as the source PDF file.

5.2.3. Running sample files (GCC)

Sample files for Linux and macOS distribution are located in the
`/opt/pdf2xml/samples/c++/` directory.

In order to compile a Sample file with gcc you only need to specify directories of shared libraries and main header file. When current directory is the location of sample file you can use the following command:

```
gcc Sample.cpp -I"../../src/c++/" -I"../../include"
```

After the successful compilation you can run the executable by providing at least two arguments, consisting of the absolute path to the input PDF file and path to the converted output file.

> ***Note:*** In order for the output file to be properly interpreted by other applications, it is strongly recommended for the output file extension to be .xml. You also have the ability to change (or add) the extension after the conversion is complete.

Once you have successfully converted the file using the Sample application, you will see the confirmation in the Command Line window.

If no error occurred, the process will terminate automatically with an exit code 0. Output formatting can be changed by passing on different **`enum class PDF2XML::Formatting`** value to the conversion function.

5.3. Net C# sample project

This section is intended to simplify connecting the PDF2XML.NET.dll proxy library for Visual C# Sample project. First things first, you need to locate the sample project. This sample project is placed by default in the Documents directory, expressed through an environment variable in the location of:

`"%USERPROFILE%\Documents\PDF2XML Samples\Net.C#"`

It is strongly recommended to make a copy of Sample files on which you intend to work on, as this will make it easier to rollback changes if the need arises.

5.3.1. Resolving Dependencies

The first step in configuring sample files to release the power of PDF2XML.dll is running the "Sample.sln" file located in the ".`\Net.C#"` directory. This can be accomplished either by using Open Project/Solution from the Visual Studio menu, or by simply double clicking on Sample.sln file.

When the Solution is opened, no project files will be loaded at start, which is normal. However, by simply opening the Sample.sln solution, you'll see that Visual Studio has prepared additional directories required for successful running of the sample application, specifically `"bin"` and `"obj"` directories, placed in the same directory as the Sample.sln file.

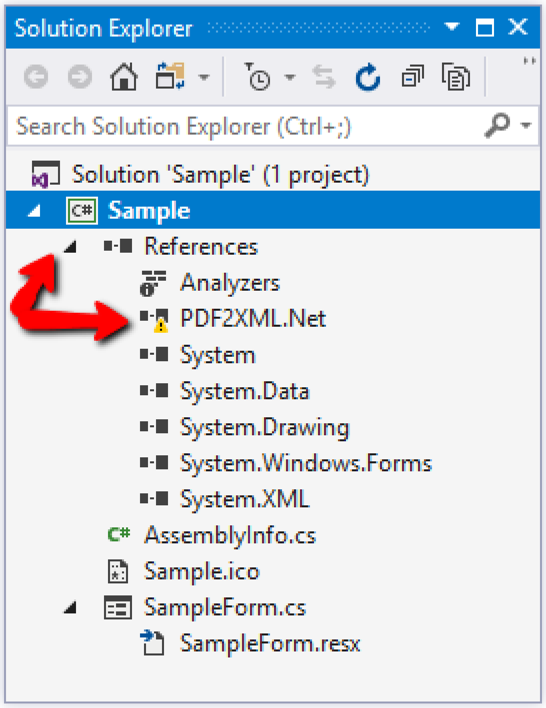
The directories that are of interest here are the sub-directories of the `"bin"` directory, which will store executables after the first build of the project. The name of the directory will depend on the Build Configuration used to build the application. Since the default configuration is "Debug" we will refer to this directory as the target one.

Next, using Windows Explorer, navigate to the installation directory of PDF2XML SDK, select all files and folders in the installation directory, and copy them in the `./bin/Debug/` directory of your sample project.

Once the files have been copied, you need to check if the proxy library PDF2XML.Net.dll dependencies have been resolved. Look for the Solution Explorer Window in your Visual Studio 2007. If Solution Explorer is not present, you can open it from the main menu: View -> Solution Explorer.

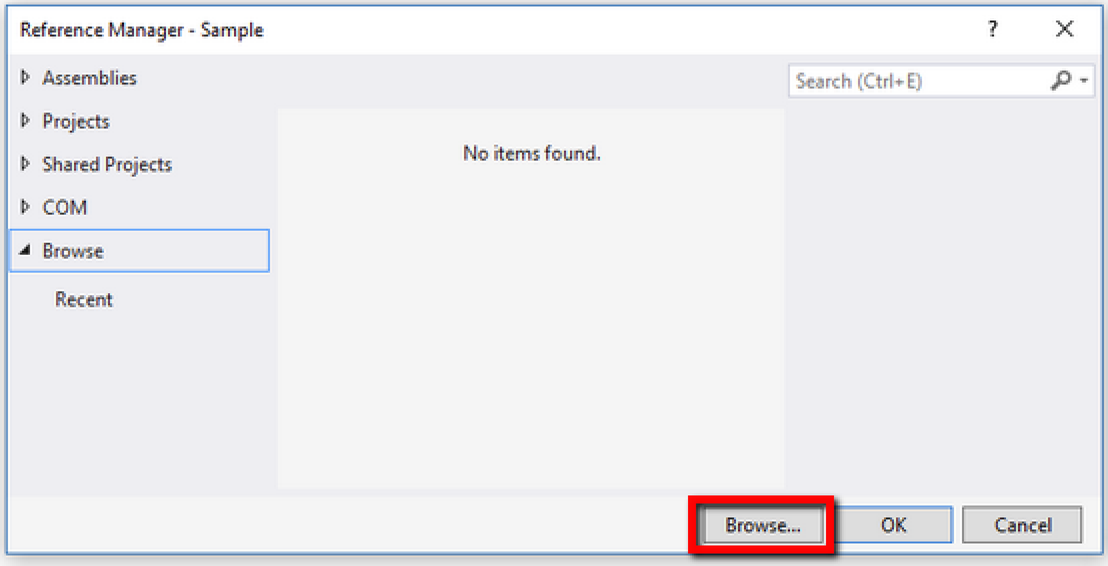
Inside Solution Explorer Windows, click the arrow-triangle left of "References" to see current project references.

Solution Explorer should look similar to this:



If marked by a warning exclamation mark, PDF2XML.Net dll is not properly linked. This situation is shown in the screenshot above.
To fix this, right click on "References" to bring up context menu, and choose "Add Reference"

You should get the following window:



Note: It is quite common for the list of Recent files to be populated with other files if you worked with Visual Studio before. To avoid unnecessary clutter this list has been cleared, as it is shown in the screenshot.

On this screen click on the Browse… button and navigate to the location of your PDF2XML SDK tools. This is usually the installation directory chosen during the installation. Once you have navigated to the location of PDF2XML SDK tools, choose and add PDF2XML.Net.dll. After that, simply confirm the selection by clicking on the OK button.

5.3.2. Analyzing the SampleForm.cs

Once the References are resolved, you can open the `SampleForm.cs` from the solution explorer.

Because the SampleForm class is a subclass of `System.Windows.Forms.Form`, it will open by default in *design mode*. In order to access the code, you can either double click on design window, or simply press the F7 keyboard button, while `SampleForm.cs` is selected in Solution explorer.

With SampleForm.cs code visible, scroll down to main function. It should have the following structure:

```
static void Main()
      {
            PDF2XML.Initialize("XXXXXXX");
            Application.Run(new SampleForm());
            PDF2XML.UnInitialize();
      }
```

Argument of the method **`public static void Initialize(string pass)`** needs to be changed in order to use PDF2XML as a licenced product. This method accepts a String value that should be your personal password which is one of the two parts required to have a licenced SDK. For more details refer to Chapter 3.

Here, the password has been replaced with "XXXXXX". To successfully run the sample application in non - trial mode, you will need to pass in the password provided after purchasing the licence.

Next, we need to concentrate on the function **`okButton_Click()`** which is triggered when the user clicks on the OK button on the Form. This part of the Sample application is controlling the actual call to conversion function.

Note: The current Sample solution doesn't automatically add an extension to the file names. It is up to the developer to set appropriate file extensions if needed.

You can control the output formatting of your xml files. The following code snippet provides the logic behind the drop-down menu of the form:

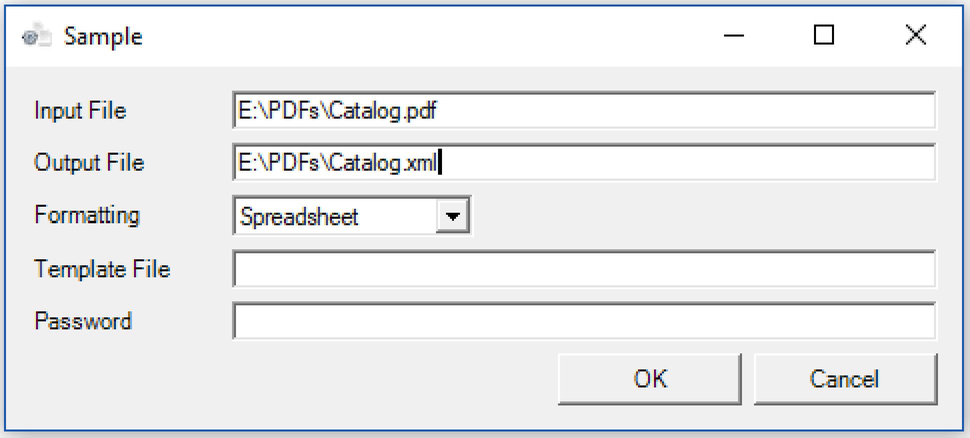
```
switch (formattingComboBox.SelectedIndex)
           {
           case 0: formatting = PDF2XML.Formatting.Document;
                    break;
           case 1: formatting = PDF2XML.Formatting.Spreadsheet;
                    break;
          }
```

The chosen formatting will have an effect on how data in the output file will be formatted.

Enum Value	Formatting	Value
`Formatting.Document`	Document	1
`Formatting.Spreadsheet`	Spreadsheet	2

5.3.3. Running the Sample Application

After you've successfully built the application and gave it a good test run, you should see the following window:



Input File: E:\PDFs\Catalog.pdf
Output File: E:\PDFs\Catalog.xml
Formatting: Spreadsheet
Template File:
Password:

The last step needed before the PDF2XML engine can be seen in action is to supply at least two fields in the running form: Input File and Output file.

After the file paths have been provided, simply click on OK and after some time, depending on the size of the file, you'll have your first converted file using PDF2XML.

Note: If you're running a trial version of the SDK, a maximum of three pages of the input .pdf file will be converted, and a notice will be given about trial limitations inside the converted file.

The password field is mandatory only if the Input File is password protected. Currently, only ASCII / ISO 8859-1 (Latin-1) characters are supported for passwords.

5.4. Python Module

This section explains how to set up a Python proxy module to perform conversions using Python programming language. The Python interface is accessing the API of native Shared Libraries through the usage of ctypes .

Python modules and sample files are deployed as open source documents. There are two proxy modules provided:

❏ **PDF2XML.py**
❏ **PCS.py**

On windows these are located in the `<InstallDir>/Sources/Python` directory . The sample project itself is placed by default in the Documents directory, in the location of:

`"%USERPROFILE%\Documents\PDF2XML Samples\Python"`

It is strongly recommended to make a copy of Sample files on which you intend to work on, as this will make it easier to rollback changes if the need arises.

Due to its flexibility, Python Sample files are deployed without project files giving the user freedom to choose the tool for the testing. For the most universal approach this document will explain the usage through the bare command line interface. It is recommended to use Python version 3.6.6 or greater when running Sample files.

5.4.1. Resolving Dependencies

If you choose to leave the task of setting up Path variables automatically during the installation process (or they have been manually added), all the dependencies for Python Sample files should be immediately resolved.

Both Sample files and Proxy modules have pre-set path resolution to ensure seamless and quick integration to your environment. If the search for either a proxy library or native shared libraries fails, you have two options: either modify the search logic to meet your specific requirements, or put all of the necessary resources in the same directory.

The straightforward quick-fix in case of unresolved dependencies would be to move both proxy modules and Sample files to the location of the Shared library which is the `.\Sources\Python\` of the install directory for Windows, or `./bin` directory on Linux and macOS.

5.4.2. Running the Sample module

Sample modules are pre configured so that they can be run directly in trial mode without requiring the user to make any changes to the code. Simply run the Sample.py module with python, providing paths to the Input and Output file names as arguments.

Example: The following command on windows would convert file named *Tax.pdf* into XML file with the same name:

```
> py Sample.py F:\PDFs\Tax.pdf F:\Output\Tax.xml
```

This assumes that your current working directory (pwd) is the location of the Sample.py file, and that python shorthand variable is set to py.

It is also possible to load Sample.py into your favorite IDE and run from there. You only need to either provide command line arguments, or hard code paths to the Sample files themselves.

> **Note:** While running the module in trial mode doesn't require any changes to the Sample files themselves, when you purchase the Licence from Investintech.com you might need to change the parameter passed to Initialize() method. You can read more about licencing in Section 3.

When running a trial version of the SDK, a maximum of three pages of the input .pdf file will be converted, and a notice will be given about trial limitations at the end of the converted file.

5.5. Java Sample Project

This section explains how to quickly set up a Java proxy module which allows you to perform conversion using Java programming language through the proxy Shared Libraries and Java Wrapper.

Java proxy module consist out of two composite components:

- ❏ **PDF2XML.Java**[**.dll** | **.so** | **.dylib**] (Win | Linux | macOS)
- ❏ **PDF2XML.jar**

The JAR archive contains both Java class files and the source code of the Java wrapper to ensure flexibility and ease of use.

For easier initial integration .jar archive we have provided you with Sample projects.
This sample project on Windows is installed by default in the "Documents" directory, at the location of:

`"%USERPROFILE%\Documents\PDF2XML Samples\Java.Java"`

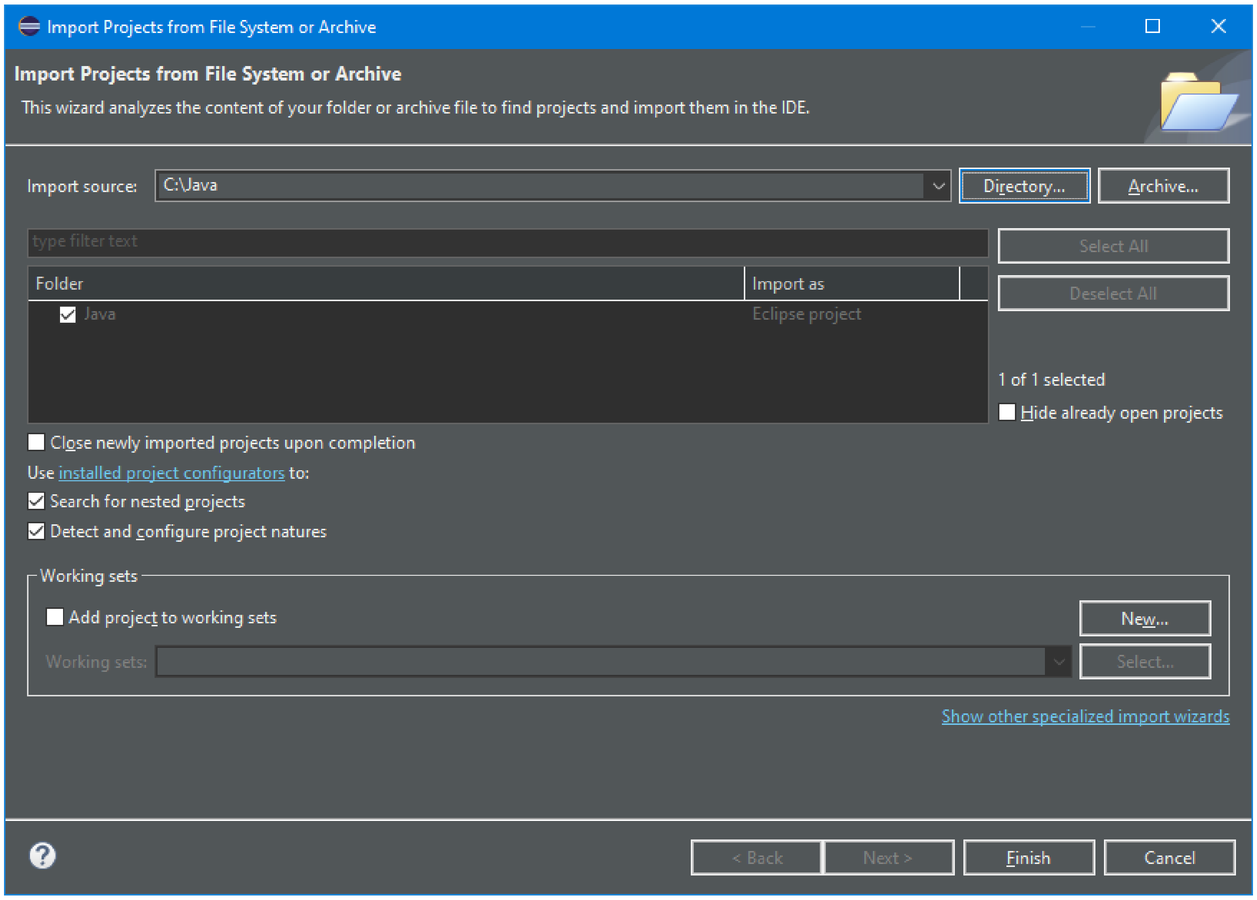
It is strongly recommended to make a copy of Sample files on which you intend to work on, as this will make it easier to rollback changes if the need arises.

5.5.1. Resolving Dependencies (Eclipse)

Sample files for Java come with a ready to load project for the Eclipse IDE. This guide will provide instructions for the integration of PDF2XML Java proxy library through the usage of Eclipse IDE (2020-09) as one of the most popular Java IDEs available.

These instructions assume that the user has already installed JDK(minimum version 8) and have successfully integrated Eclipse with the local environment.

To load the Sample project in Eclipse go to `File → Open Projects from file system...` From the Import Project Wizard click on Directory and browse to the location of the Java Sample files. After the project is imported and parsed by the IDE, simply click Finish to complete the project import.


Import Projects from File System or Archive

Import Projects from File System or Archive

This wizard analyzes the content of your folder or archive file to find projects and import them in the IDE.

Import source: C:\Java

Folder	Import as
☑ Java	Eclipse project

1 of 1 selected

☐ Hide already open projects

☐ Close newly imported projects upon completion

Use installed project configurators to:

☑ Search for nested projects

☑ Detect and configure project natures

Working sets

☐ Add project to working sets

Working sets:

Show other specialized import wizards

After closing the Import Wizard you should see the project in the Package Explorer. At this point you are most likely getting unresolved dependency warnings for the PDF2XML project. In order to resolve these dependencies, you need to include the PDF2XML.jar archive into your project.

To include External JAR containing Java Wrapper for the C++ shared library, while having project selected (or right clicking on the project) navigate to:

`Project` → **`Properties`** → `(Left List)` **`Java Build Path`** → `(Tab)` **`Library`** → `(button)` **`Add External JARs..`**

From the file browser window, navigate to the install location of PDF2XML SDK and select the .jar file to import. Once a Java Archive file has been imported you should have all Java Native dependencies resolved.

> **Note:** It is also possible to extract source code from .jar archives.If you decide to do this you need to manually include `PDF2XML.java` and `Exception.java` files into your projects.

One more dependency issue that you may encounter is when Java Wrapper is not able to find PDF2XML.java Shared library. These libraries are built separately for each OS platform and have different extensions:

❏ Windows:	PDF2XML.Java.dll
❏ Linux:	libpdf2xml.Java.so
❏ macOS:	libpdf2xml.Java.dylib

In order for the Java Virtual Machine to be able to find C++ Shared Library directory containing these libraries, your local environment should have set up the PATH variable (Win) or LD_LIBRARY_PATH (Linux). You can also explicitly assign it as a command line argument when running Sample.class, by using `-Djava.library.path=<path_to_lib>`

In order to setup Eclipse with additional path, you can the specific path to the IDE textbox at:

Run → **Run Configurations** → (Tab) **Arguments** → **VM Arguments**

Last but not least, the Proxy Library should be able to resolve the path to the natvie library. Proxy libraries will search for the nativ lib in the same directory where they are placed and in directories specified by PATH environment variables.

5.5.2. Resolving Dependencies (Terminal)

For the highest degree of compatibility, you can also use terminal to compile sample files. In order to resolve dependencies, pass on appropriate paths to classpath.

Example: On linux, assuming that your current directory is the location of Sample.java file, you can use following command to compile Sample file:

```
javac -classpath "/opt/pdf2xml/lib/pdf2xml.jar" Sample.java
```

Similar commands can be used on Windows, using OS specific paths.

5.5.3. Running the Sample application

Once you have resolved dependencies, you should be able to run Sample applications by providing input and output file paths, either as command line arguments or by hardcoding them in the Sample.java.

You can provide command line arguments through the Eclipse interface from:

Run → Run Configurations → (Tab) Arguments → Program Arguments

If you prefer, you can also run the binaries from the terminal. Keep in mind that you also need to provide a .jar archive containing wrapper Java classes for proxy libraries.

Example: You can use following command to run compiled Sample.class on Linux, assuming that the current directory is the directory of the binary file.:

```
java -Djava.library.path=/opt/pdf2xml/lib/ -classpath
"/opt/pdf2xml/lib/pdf2xml.jar" Sample <in_path> <out_path>
```

Above command assumes that you have not set LD_LIBRARY_PATH in your local environment. For that reason we are passing on the explicit path of pdf2xml.so to JVM.

After a short time, depending on the size of the PDF file, you should have your PDF file converted at the stated location.

> **Note:** While running the module in trial mode doesn't require any changes to the Sample files themselves, when you purchase the Licence from Investintech.com you might need to change the parameter passed to **`Initialize()`** method. You can read more about licencing in Section 3.

When running a trial version of the SDK, a maximum of three pages of the input .pdf file will be converted, and a notice will be given about trial limitations at the end of the converted file.

6. Appendices

6.1. A - Troubleshooting: Command Line Tool

We, at Investintech, do our best to make our applications and development tools as stable as possible. Of course, no application is errorless, and when an error does happen it can often be avoided by consulting with the following table:

Problem	**Possible Solutions**
Input file not found: <filePath/Name>	1. Use absolute file path example: "E:\PDFs\SomeFile.pdf" 2. Check the spelling 3. Use wildcard characters and/or recursion switch **-r** (section 3.4.1)
Output file exists: <filePath/Name>	1. Specify a different output Directory (section 3.4.2) 2. Delete the previously created file at the location <filePath> 3. Use **-eo** or **-em** to overwrite or modify your output file name (section 3.4.6)
Unrecognized input file format: <filePath/Name>	1. When using multiple file conversions with wildcards: 1.1. Increase restrictions on mask Example: use .pdf extension (*.pdf) 1.2. Use **-g** switch to skip error (section 3.4.9) 2. When using single file conversion: 2.1. File format is not supported..

Problem	**Possible Solutions**
SAXParseException: Tag mismatch in line 9 column 4	1. Create or choose another template file (.pcvt) 2. Re-create an Option.pcvt file in Able2Extract Desktop App
Initialization failed: 10	1. Check for password and Licence integrity
PCS::ErrorCode::wrongParam	1. Check if correct parameters are used. For reference see (Section 3.4) 2. Make sure no space is provided between a flag and the passed parameter.
The filename, directory name, or volume label syntax is incorrect.	1. Make sure that the first argument is either the correct path to a file name, or a correctly formulated wildcard combination, followed by the **-i** switch with the correct folder location.
PCV::fileCreate Function: Investintech::PCV::File::File,Line: 40	1. Delete/more/rename output resource folder if it collides with file name

6.2. B - Troubleshooting: Shared Library

The table below contains the most common problems which users encounter when using Sample files. For a full list of possible errors, please refer to header file LIB.h, which can be used for reference when the conversion error is given in row value.

Problem	**Possible Solutions**
System.DllNotFoundException: 'Unable to load DLL 'PDF2XML.dll'	1. Make sure that the required files and directories are in the same directory as binaries (see section 5.2.1)
Warning LNK4272 library machine type 'x64' conflicts with target machine type 'x86'	1. Change your specified CPU platform architecture build from Configuration Manager in VS Section 5.2.1 2. Download the appropriate SDK build
Conversion failed: 3024 (File Save error)	1. Change your output file name 2. Delete the previous output file
Conversion failed: 5 Unrecognized File Type	1. Check the input file name for spelling errors 2. Include proper file extensions of the input file name 3. Choose .pdf file for input file
Conversion failed: 2 Wrong Param	1. Check the parameters passed 2. Check if format and formatting are compatible (Section 5.2.4)
Conversion failed: 3022 File Create	1. Change output file name 2. Check for Resource folder naming collision, do one of the following: delete, rename or move resource folder in the output dir.

Appendix C - Class and method Description

PDF2XML Proxy libraries public API consists of three static methods, a Document class containing Constructor used for opening the files, and five operational methods for operations to be performed on loaded files.

Summary of static class methods:

```
● static void Initialize( String password )
```

Perform initialization using a personalised password.

Must be first to run before any operational methods.

password -- Personalised password, which is the first licencing component. The second part is the Licence.lic file that should reside in the tools directory.

```
● static void UnInitialize()
```

Perform Unintialization to conserve system resources when engine is no longer needed.

```
● static void Convert( String inFilePath,
                       String outFilePath,
                       int formatting,
                       String templateFilePath,
                       String filePassword )
```

Convert PDF file passed as inFilePath to destination outFilePath

inFilePath -- Path to the input PDF file.
outFilePath -- Path to the output file.
formatting -- The output formatting of the converted file.
templateFilePath (optional) -- Path to the template Options .pcvt file. **filePassword** -- Required for encrypted PDF files. Pass null if it is not required.

Instantiation of **Document** class is used to open PDF files.

```
●  Document(   String inFilePath,
               String templateFilePath,
               String filePassword )
```

For a more precise conversion, this class needs to be instantiated. Each object of this class holds exactly one PDF file. The corresponding functions can be used to reduce the PDF conversion to a specific page or page range.

inFilePath -- Path to the PDF file intended for conversion.
templateFilePath -- Path to the template Options.pcvt file.
filePassword -- Password for encrypted documents.

File operations can be performed on the instances of the Document class:

```
●  void Convert(     String outFilePath,
                     int formatting,
                     String templateFilePath)
```

Convert PDF file passed as inFilePath to destination outFilePath

outFilePath -- Path to the output file.
formatting -- The output formatting of the converted file.
templateFilePath (optional) -- Path to the template Options .pcvt file.

```
●  int GetPagesNum()
```

Read the number of pages of the PDF file.
return integer -- Number of pages in the file.

```
●  void SelectPage(int pageNo)
```

Select (mark) a single page for conversion.
pageNo -- Page number to be selected for conversion.

```
●  void SelectPageRange(int firstPageNo, int lastPageNo)
```

Select (mark) multiple pages for the conversion.
firstPageNo --Number of first page to be converted (inclusive).
lastPageNo -- Number of last page to be converted (inclusive).

- `void` **`RemoveSelection`**`()`

Remove (unmark) selection for the current document.

Calling this method sets your selection to the default selection value which is all pages.
Calling the Convert() method after removing all selection will convert the whole document.

- `void` **`Close`**`()`

Close the PDF file associated with this object.