

# PDF to Excel

Software Development Kit



**INVESTINTECH.COM**  
P D F S O L U T I O N S

Copyright 2019 Investintech.com, Inc. All rights reserved

Excel, Windows, Visual Basic, Visual C++, Visual C#, Visual Studio, .Net, PowerShell, Win32, Windows NT and Windows 10 are either Trademarks and/or Registered Trademarks of Microsoft Corporation (MS).

Adobe, Acrobat and Postscript are either Trademarks and/or Registered Trademarks of Adobe System Incorporated.

## **Contents:**

<b>1. Introduction</b>	<b>4</b>
1.1. System Requirements	4
1.2. Getting more information	5
<b>2. Installing PDF to Excel SDK tools</b>	<b>6</b>
<b>3. Registration</b>	<b>11</b>
3.1. Using Licence file	11
3.2. Using Password to activate licence	11
<b>4. Using the Command Line Tool</b>	<b>12</b>
4.1. Preparation	12
4.1.1. Adding Path variable (Optional)	13
4.2. Quick Single File conversion	14
4.3. Quick Multiple File Conversion	15
4.4. Command line arguments	16
4.4.1. Specifying input directory	19
4.4.2. Specifying output directory	20
4.4.3. Output file format	23
4.4.4. Page ranges	24
4.4.5. Using template files	26
4.4.6. File name collision options	27
4.4.7. Verbosity level of Console output	29
4.4.8. Converting password protected files	30
4.4.9. Ignoring Errors	31
4.4.10. Parallel Conversion	32
<b>5. Using the Shared Library</b>	<b>34</b>
5.1. Preparation	34
5.1.1. Licence Registration	34
5.2. Lib.c sample project	35
5.2.1. Resolving Dependencies	37
5.2.2. Running the sample project	38
5.2.3. Registration	40
5.2.4. Choosing different output format	40
5.3. Net C# sample project	41
5.3.1. Resolving Dependencies	42
5.3.2. Analyzing the SampleForm.cs	44
5.3.3. Running the Sample Application	45

<b>6. Conversion options for template Files</b>	<b>47</b>
6.1. Command Line Tool: Options.pcv	47
6.2. Shared Library Options.pcv	49
6.3. Combining template file and options file	49
<b>7. Appendices</b>	<b>50</b>
7.1. A -Troubleshooting: Command Line Tool	50
7.2. B - Troubleshooting: Shared Library	52

# 1. Introduction

The main purpose of this document is to provide you with a detailed guide on how to get started with the PDF2Excel Command Line Tool and how to use Sample Files for developing an application using the PDF2Excel Shared library.

PDF to Excel SDK is a complete package that comes with the following components:

- ☐ PDF2Excel Command Line tool
- ☐ Shared resources library
- ☐ Sample files

## 1.1. System Requirements

The following are the requirements for using the PDF2Excel SDK tools comfortably:

- ☐ Windows 7, Windows 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

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

---



### Technical support

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

---



### Fax and Mailing Address

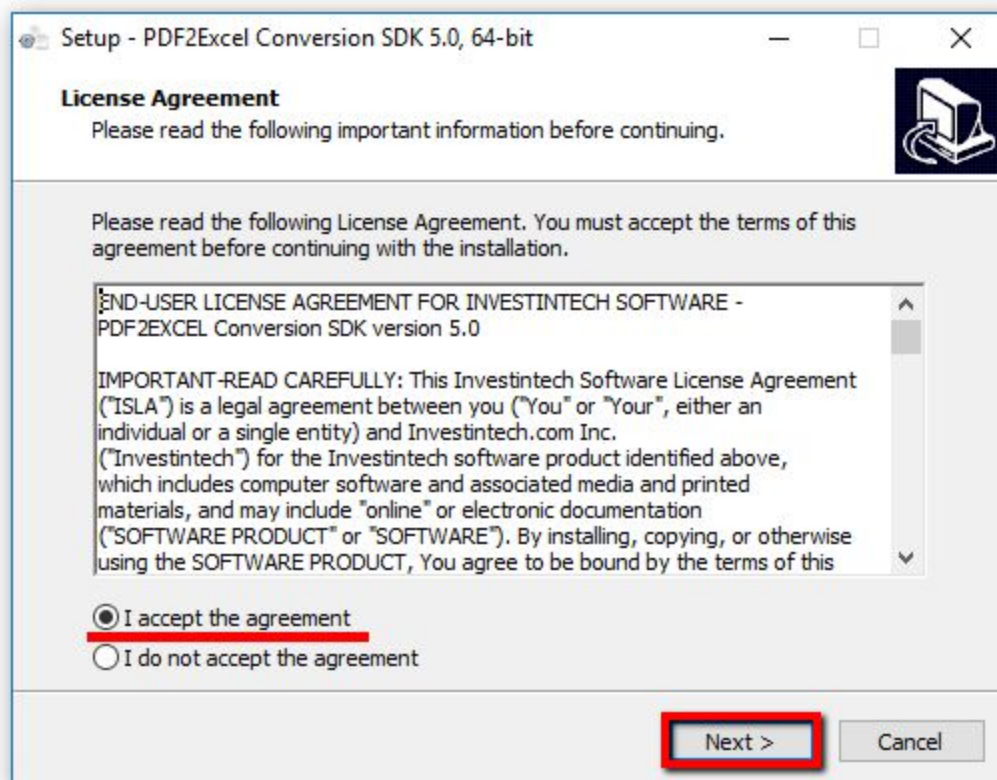
<b>Fax:</b>	+1 416 920 5848
<b>Mail:</b>	Investintech.com, Inc. 301 - 425 University Avenue Toronto, ON Canada M5G 1T6

## 2. Installing PDF to Excel SDK tools

Thank you for choosing PDF to Excel SDK tools from Investintech.com, Inc.

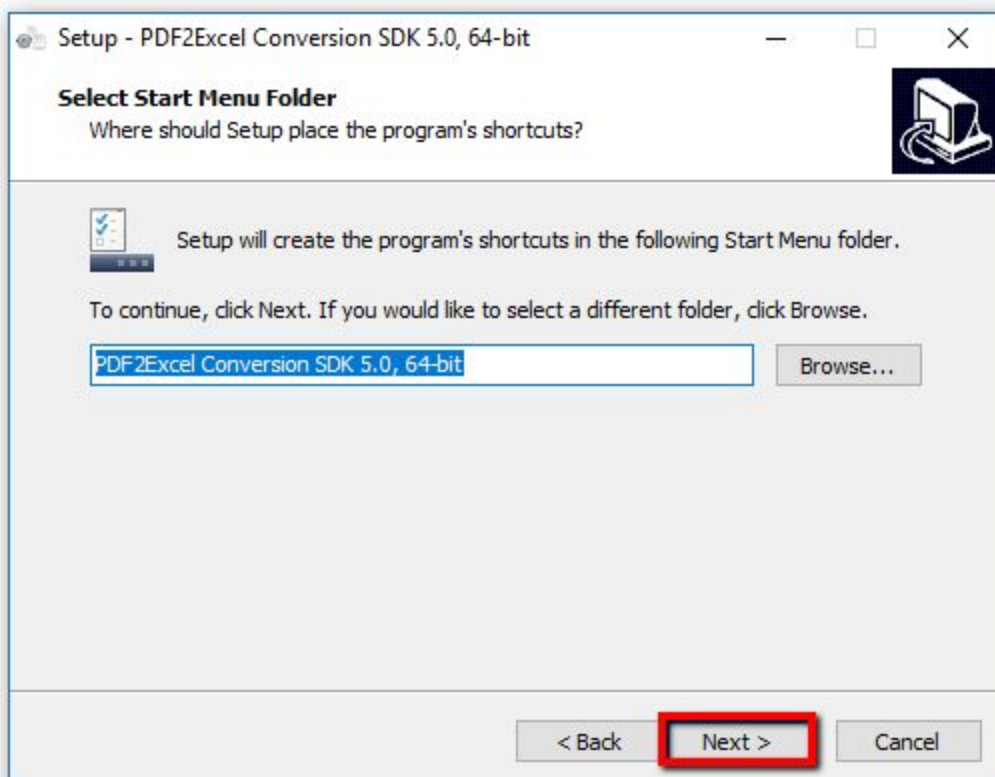
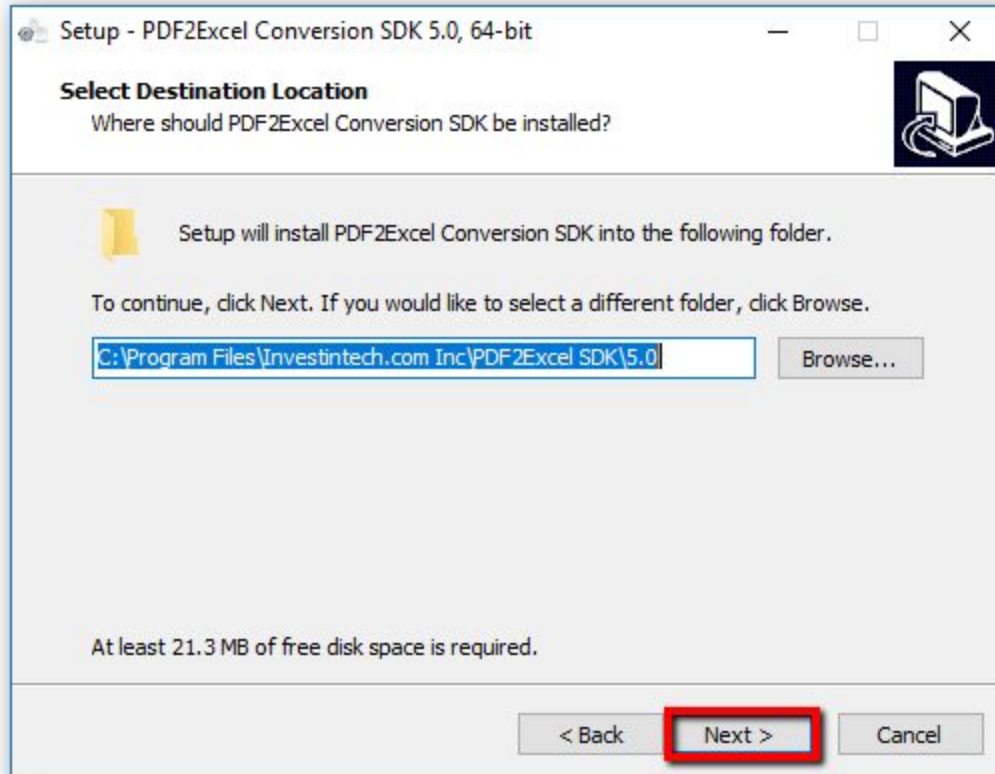
By following the steps below, you can quickly start using the PDF2Excel SDK tool of your choice.

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:



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 PDF to Excel 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 as displayed in the screenshots below:





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.

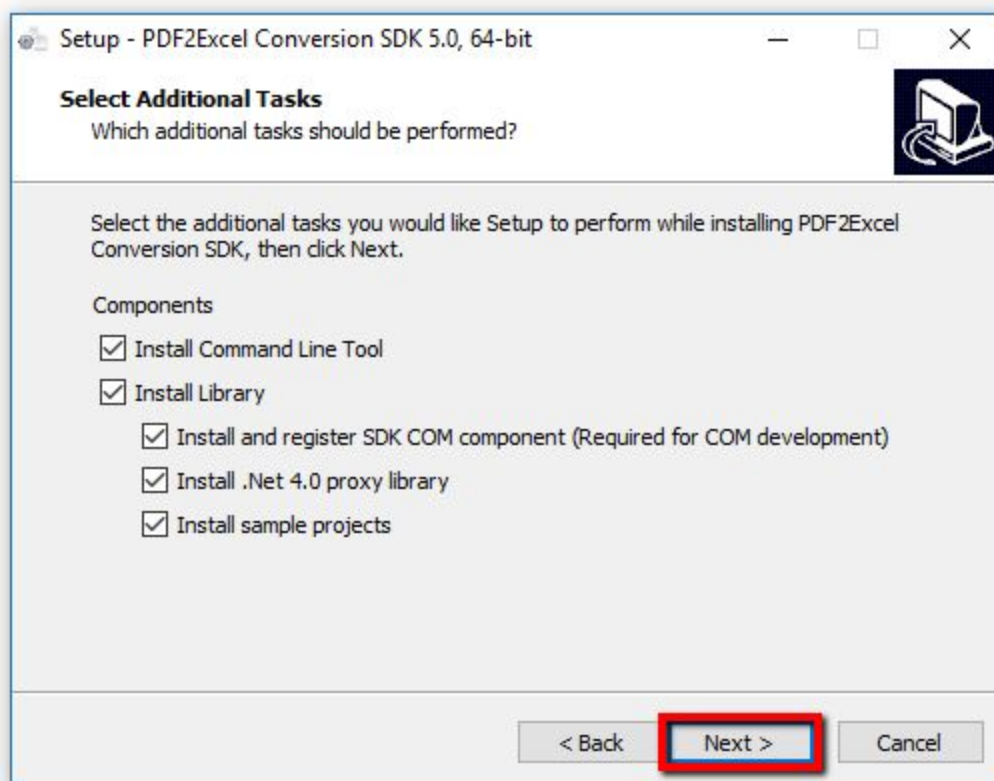
There are two major components of the SDK tools:

- ☐ Command Line Tool (CLT)
- ☐ Shared Library

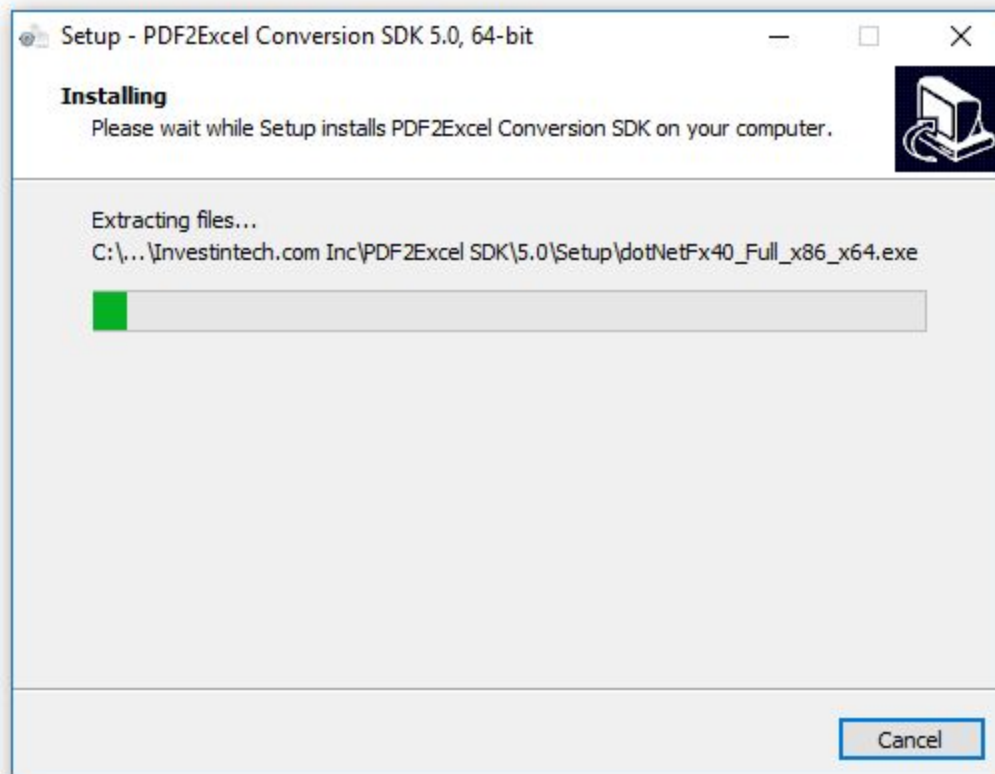
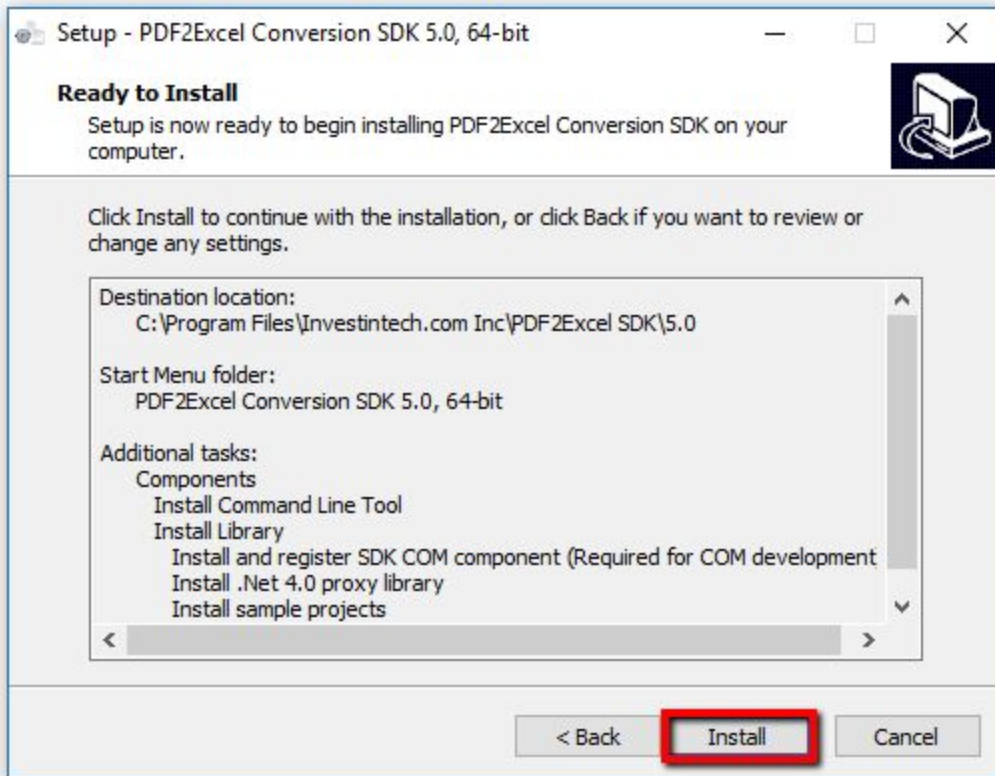
The Command Line tool usage is discussed in detail in Chapter 3 and the PDF to Excel Library usage is discussed in Chapter 4.

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 PDF to Excel SDK tools also come with Sample Projects for Visual Studio. These sample projects will be copied to the "Documents" directory or more precisely, to the following location on a local machine: **%USERPROFILE%\Documents\PDF2Excel Samples**



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 PDF to Excel SDK!

## 3. Registration

If you decide to purchase our PDF2Excel 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](http://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

To use the personalized License.lic file with either the Command Line Tool or Shared Libraries, simply copy the file in the same directory as the executables (binaries).

**Note:** When opened with any text editor, the licence file will contain your 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.

### 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);
```

**Note:** Call to the native C++ function returns `PCSErrorCode` which is an alias of type `integer(int)` . This type is defined in `Lib.h` file.

## 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.

PDF2Excel Command line tool (CLT) comes with four possible output formats for conversions:

- ☐ XLSX - Microsoft Excel 2007 and later spreadsheet (This is the default option)
- ☐ XLS - Microsoft Excel 2.0 spreadsheet
- ☐ DSV - Delimiter separated value (also referred to as CSV)
- ☐ ODS - Open/Libre Office Calc 3.0 Spreadsheet

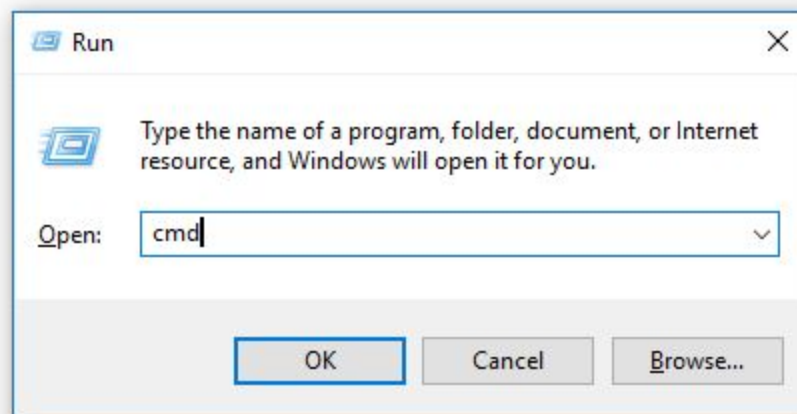
More about file formats, or specifically how to change from the default one, will be explained in [Section 4.4.3](#)

### 4.1. Preparation

In order to run the PDF to Excel 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

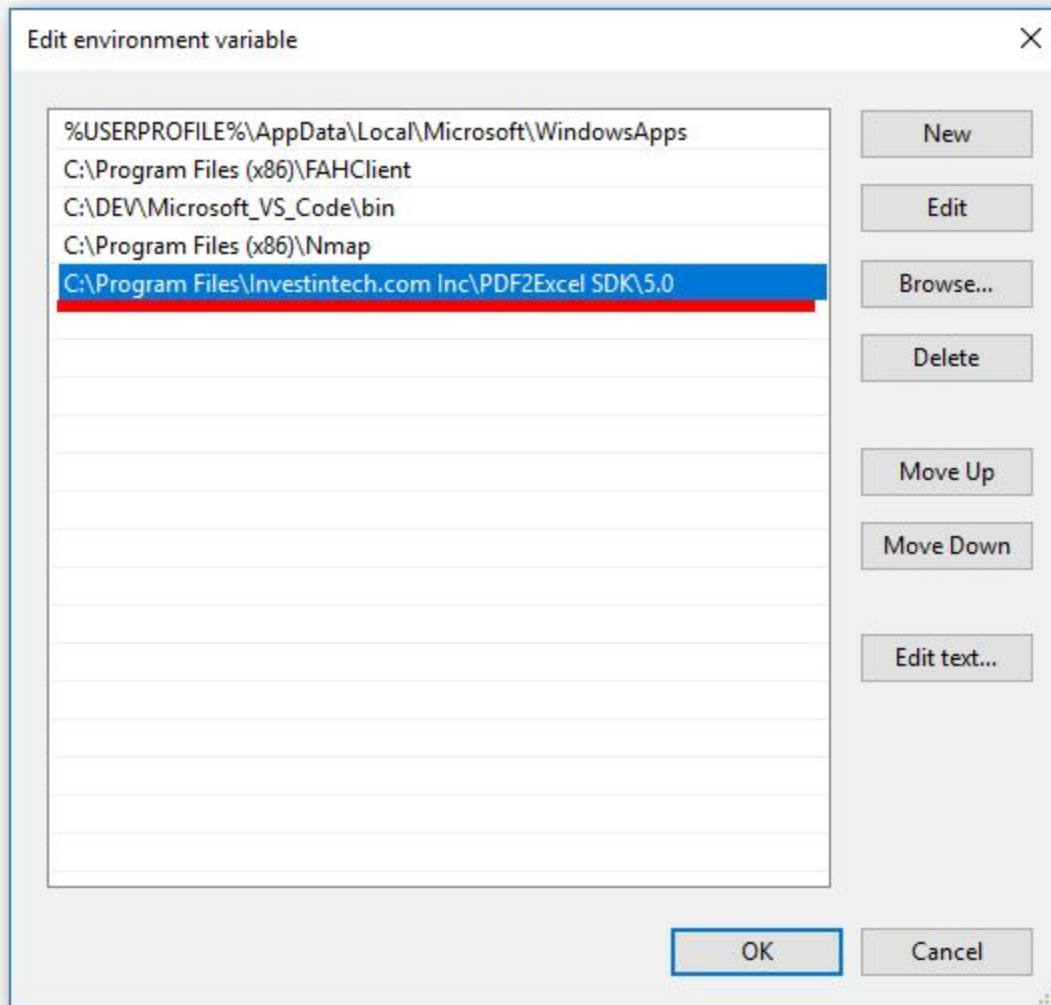


- Using File Explorer to navigate to the SDK installation directory and double-clicking on "Cmd.cmd" file. The installation directory is the one specified during installation.

#### 4.1.1. Adding Path variable (Optional)

If you would like to avoid entering an explicit path location of the PDF2Excel Console Application, you can add the location 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. In the screenshot below, the environment variable Path is referring to the default install location of PDF to Excel SDK.



## 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 *PDF2Excel.exe* with an absolute file path as your first parameter.

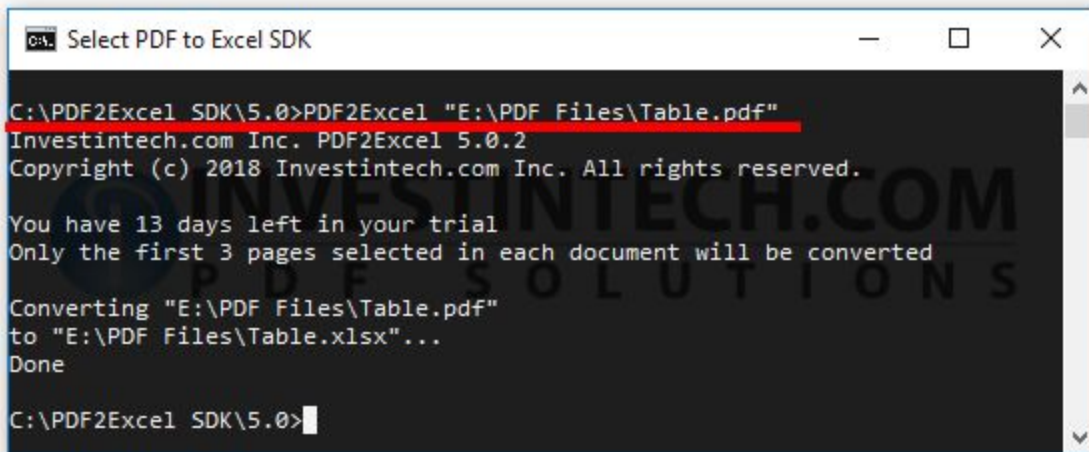
**Note:** In order for the System to find the correct executable make sure your current location in the Command Prompt is the installation directory of PDF to Excel SDK. Optionally, you could setup Environment Variables (refer to the [Section 4.1.1](#))

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

**Example:** To convert the file named "Table.pdf" in the directory "E:\PDF Files" you can issue the following command:

```
>PDF2Excel "E:\PDF Files\Table.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:\PDF2Excel SDK\5.0



```

C:\PDF2Excel SDK\5.0>PDF2Excel "E:\PDF Files\Table.pdf"
Investintech.com Inc. PDF2Excel 5.0.2
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:\PDF Files\Table.pdf"
to "E:\PDF Files\Table.xlsx"...
Done
C:\PDF2Excel 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 is the same as the input directory, which in this example is "E:\PDF Files"

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.

## 4.3. Quick Multiple File Conversion

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

The PDF to Excel 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

**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:

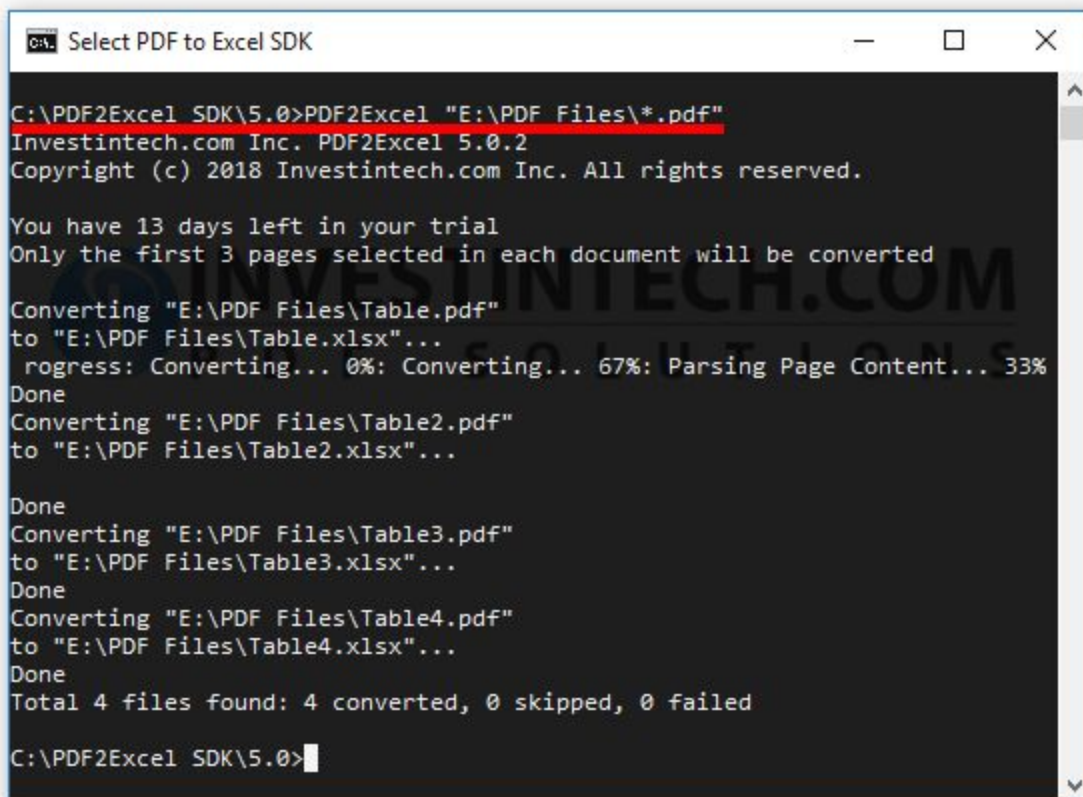
\*???pdf

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:\PDF Files" you can simply issue the following command:



```
> PDF2Excel "E:\PDF Files\*.pdf"
```



```
C:\PDF2Excel SDK\5.0>PDF2Excel "E:\PDF Files\*.pdf"
Investintech.com Inc. PDF2Excel 5.0.2
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:\PDF Files\Table.pdf"
to "E:\PDF Files\Table.xlsx"...
Progress: Converting... 0%: Converting... 67%: Parsing Page Content... 33%
Done
Converting "E:\PDF Files\Table2.pdf"
to "E:\PDF Files\Table2.xlsx"...
Done
Converting "E:\PDF Files\Table3.pdf"
to "E:\PDF Files\Table3.xlsx"...
Done
Converting "E:\PDF Files\Table4.pdf"
to "E:\PDF Files\Table4.xlsx"...
Done
Total 4 files found: 4 converted, 0 skipped, 0 failed

C:\PDF2Excel 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 "Table" string, followed by exactly one other character.

```
> PDF2Excel "E:\PDF Files\Table?.pdf"
```

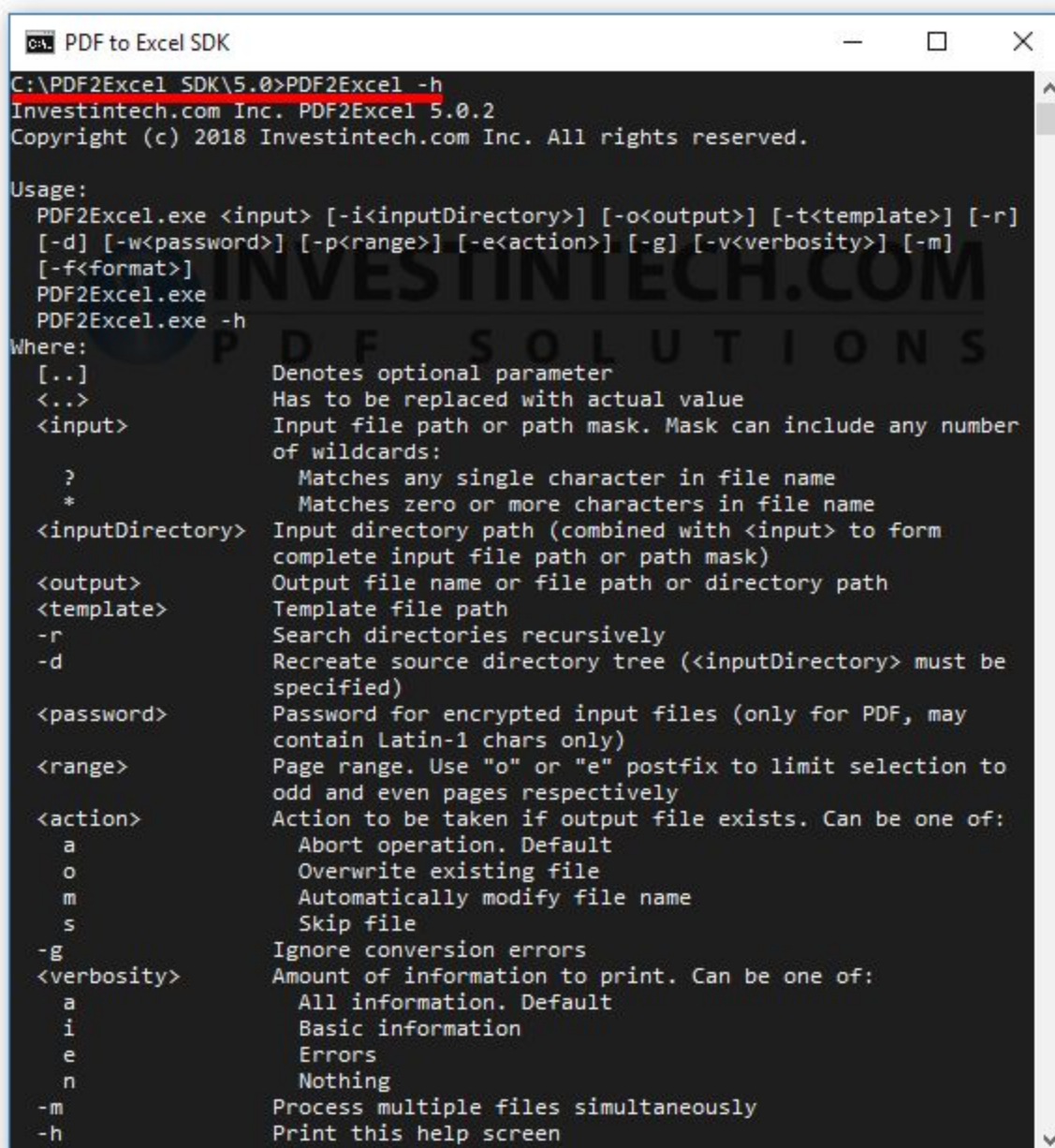
In the following chapters you will get familiar with other methods of converting multiple files using additional Command Line Arguments with the combination of wildcard characters.

## 4.4. Command line arguments

PDF to Excel 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 PDF2Excel Console Application with the **-h** switch:

```
> PDF2Excel -h
```

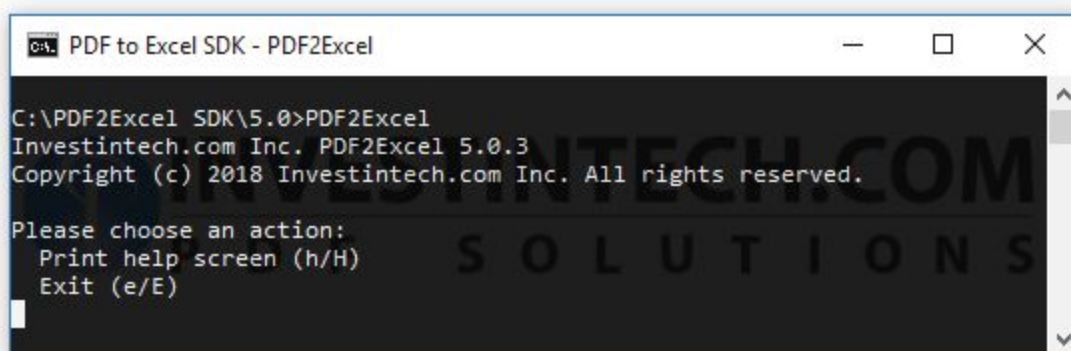


```
C:\PDF2Excel SDK\5.0>PDF2Excel -h
Investintech.com Inc. PDF2Excel 5.0.2
Copyright (c) 2018 Investintech.com Inc. All rights reserved.

Usage:
  PDF2Excel.exe <input> [-i<inputDirectory>] [-o<output>] [-t<template>] [-r]
  [-d] [-w<password>] [-p<range>] [-e<action>] [-g] [-v<verbosity>] [-m]
  [-f<format>]
  PDF2Excel.exe
  PDF2Excel.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
```

It is also possible to display help text from a simple interactive menu by running PDF2Excel.exe without any arguments, and then pressing "h" or "H".



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

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

The switch parameters that are not required are marked by NONE in 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:\PDF Files", you could use the following command:

```
>PDF2Excel * -i"E:\PDF Files"
```

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.

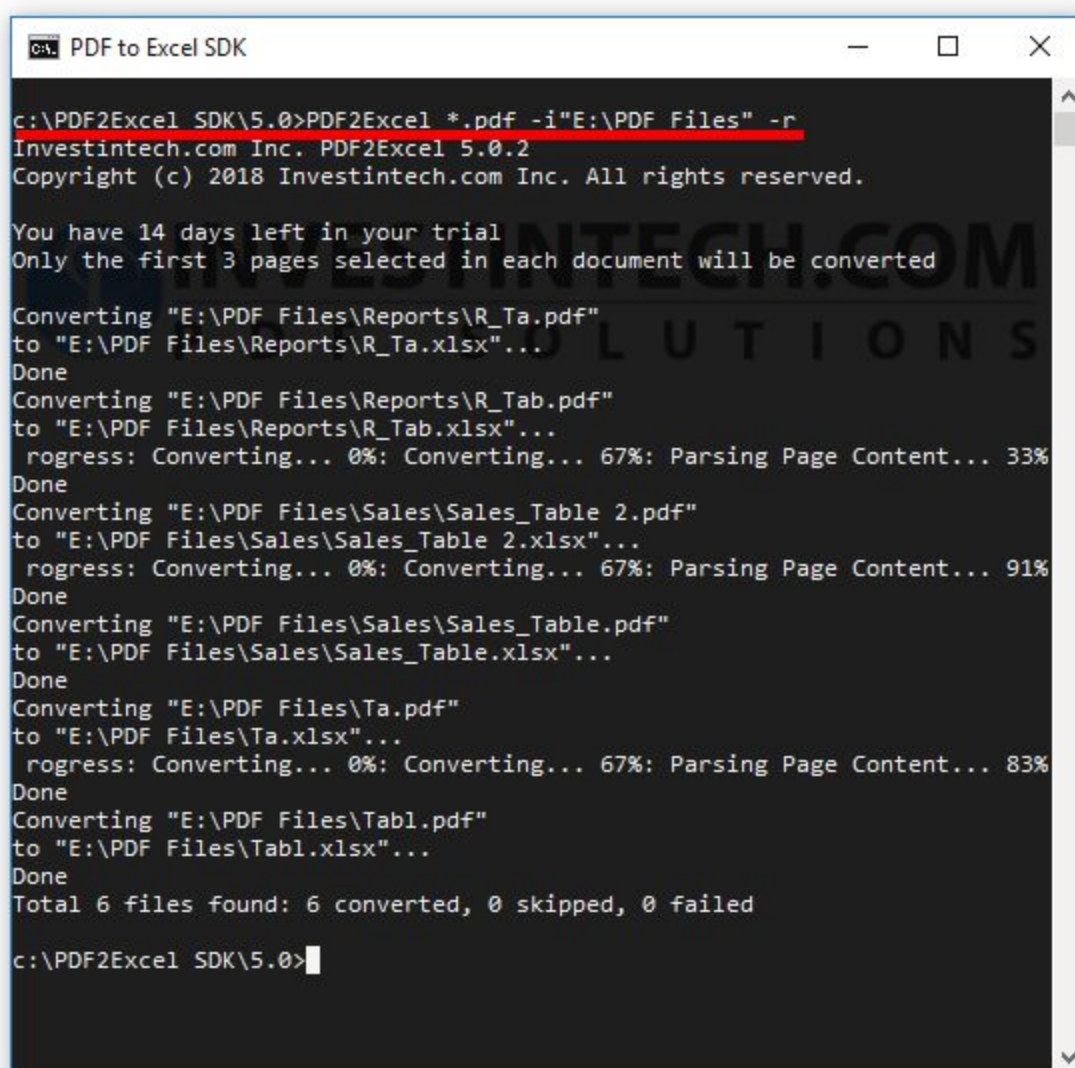
It is quite common that there are nested directories inside the specified input directory. In this case, if there are also files inside those nested directories that need to be included in the conversion output, you can use the **-r** switch to instruct the engine to perform a recursive search of files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

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

```
>PDF2Excel *.pdf -i"E:\PDF Files" -r
```

The screenshot below shows the conversion result of the following tree hierarchy of the "E:\PDF Files" 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:\PDF2Excel SDK\5.0>PDF2Excel *.pdf -i"E:\PDF Files" -r
Investintech.com Inc. PDF2Excel 5.0.2
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:\PDF Files\Reports\R-Ta.pdf"
to "E:\PDF Files\Reports\R-Ta.xlsx"...
Done
Converting "E:\PDF Files\Reports\R-Tab.pdf"
to "E:\PDF Files\Reports\R-Tab.xlsx"...
Progress: Converting... 0%: Converting... 67%: Parsing Page Content... 33%
Done
Converting "E:\PDF Files\Sales\Sales_Table 2.pdf"
to "E:\PDF Files\Sales\Sales_Table 2.xlsx"...
Progress: Converting... 0%: Converting... 67%: Parsing Page Content... 91%
Done
Converting "E:\PDF Files\Sales\Sales_Table.pdf"
to "E:\PDF Files\Sales\Sales_Table.xlsx"...
Done
Converting "E:\PDF Files\Ta.pdf"
to "E:\PDF Files\Ta.xlsx"...
Progress: Converting... 0%: Converting... 67%: Parsing Page Content... 83%
Done
Converting "E:\PDF Files\Tab1.pdf"
to "E:\PDF Files\Tab1.xlsx"...
Done
Total 6 files found: 6 converted, 0 skipped, 0 failed

C:\PDF2Excel SDK\5.0>
```

The PDF2Excel 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

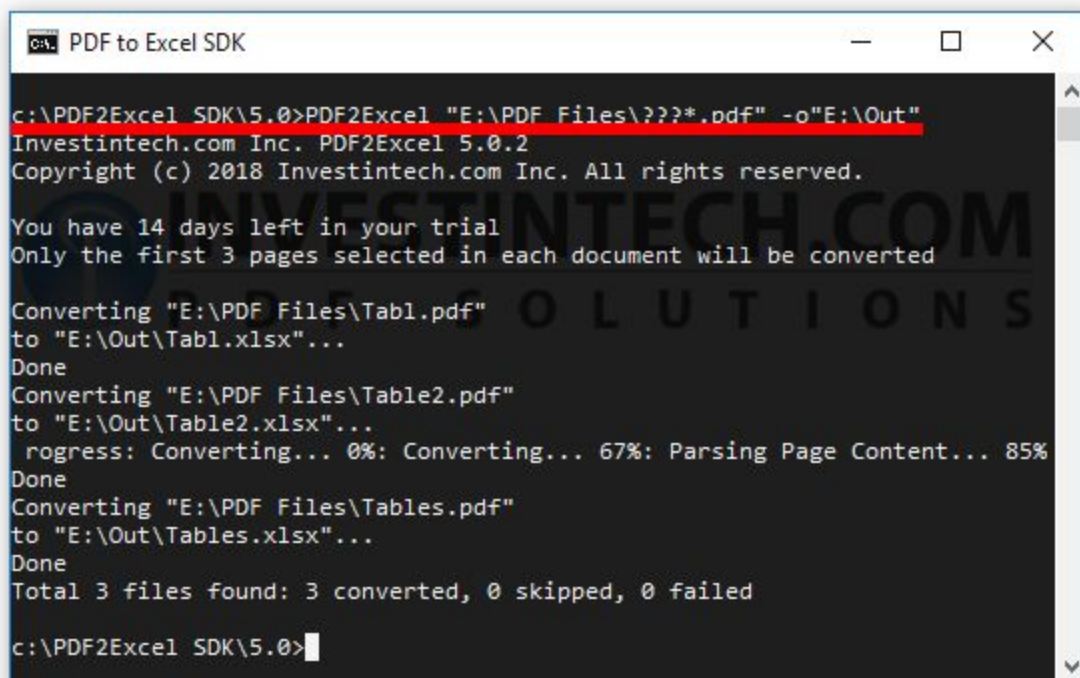


same directory as the source files, which can quickly become inconvenient as the number of source file grows larger.

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:

```
>PDF2Excel "E:\PDF Files\???.pdf" -o"E:\Out"
```



```
c:\PDF2Excel SDK\5.0>PDF2Excel "E:\PDF Files\???.pdf" -o"E:\Out"
Investintech.com Inc. PDF2Excel 5.0.2
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:\PDF Files\Tab1.pdf"
to "E:\Out\Tab1.xlsx"...
Done
Converting "E:\PDF Files\Table2.pdf"
to "E:\Out\Table2.xlsx"...
Progress: Converting... 0%: Converting... 67%: Parsing Page Content... 85%
Done
Converting "E:\PDF Files\Tables.pdf"
to "E:\Out\Tables.xlsx"...
Done
Total 3 files found: 3 converted, 0 skipped, 0 failed

c:\PDF2Excel SDK\5.0>
```

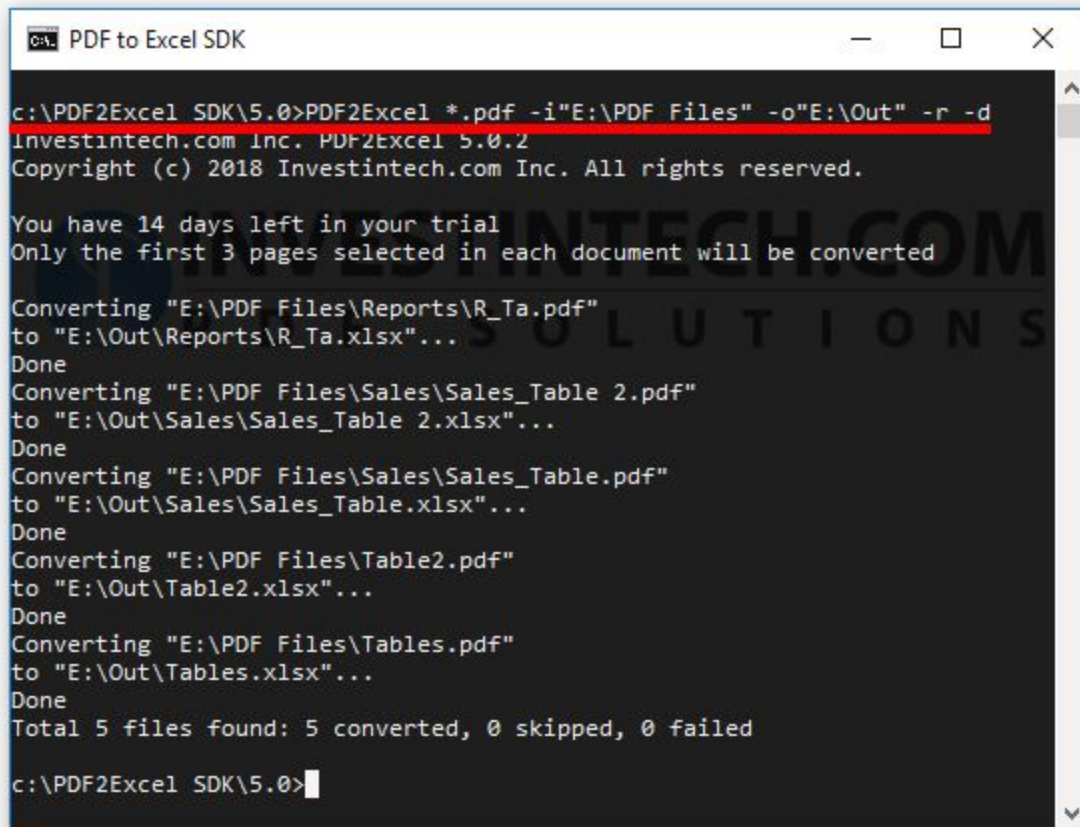
In the previous example the filter was specified for a file name using wildcards " **???.pdf** " to match only files with the extension PDF and having at least 3 characters in the filename.

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:

```
>PDF2Excel *.pdf -i"E:\PDF Files" -o"E:\Out" -r -d
```



```
c:\PDF2Excel SDK\5.0>PDF2Excel *.pdf -i"E:\PDF Files" -o"E:\Out" -r -d
Investintech.com Inc. PDF2Excel 5.0.2
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:\PDF Files\Reports\R-Ta.pdf"
to "E:\Out\Reports\R-Ta.xlsx"...
Done
Converting "E:\PDF Files\Sales\Sales_Table 2.pdf"
to "E:\Out\Sales\Sales_Table 2.xlsx"...
Done
Converting "E:\PDF Files\Sales\Sales_Table.pdf"
to "E:\Out\Sales\Sales_Table.xlsx"...
Done
Converting "E:\PDF Files\Table2.pdf"
to "E:\Out\Table2.xlsx"...
Done
Converting "E:\PDF Files\Tables.pdf"
to "E:\Out\Tables.xlsx"...
Done
Total 5 files found: 5 converted, 0 skipped, 0 failed
c:\PDF2Excel 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:\PDF Files" 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. Output file format

In the previous examples all of the pdf files were converted into MS Excel 2007 format with the extension .xlsx, which is currently the default format for the Command Line Tool.

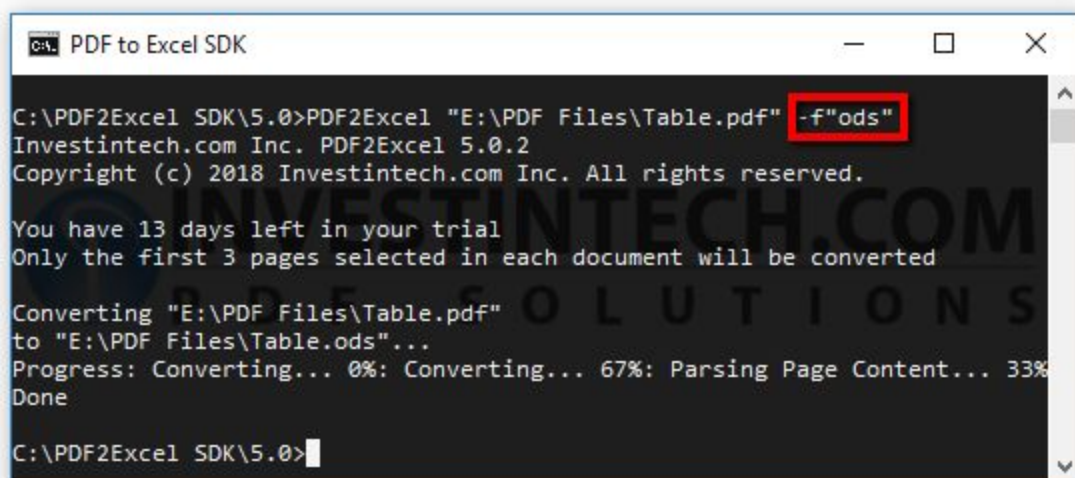
Using the **-f** switch, it is easy to change the output format to one of the other three available output file formats.

The following table below provides a list of the output formats that are supported:

-f Parameter	Format description
xlsx	XLSX - MS Excel 2007 Spreadsheet (default)
xls	XLS - MS Excel 2.1 Spreadsheet
dsv	DSV - Comma separated values, plain text, also known as CSV
ods	ODS - Open/Libre Office Calc 3.0

**Example:** To convert the "Table.pdf" file in the "E:\PDF Files" directory to the Open Document Spreadsheet format, you can issue the following command:

```
>PDF2Excel "E:\PDF Files\Table.pdf" -f"ods"
```



```
C:\PDF2Excel SDK\5.0>PDF2Excel "E:\PDF Files\Table.pdf" -f"ods"
Investintech.com Inc. PDF2Excel 5.0.2
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:\PDF Files\Table.pdf"
to "E:\PDF Files\Table.ods"...
Progress: Converting... 0%: Converting... 67%: Parsing Page Content... 33%
Done

C:\PDF2Excel SDK\5.0>
```



The double quotation marks used for the argument of the **-f** switch are not mandatory. They are only provided for the sake of brevity. So, one could have easily used:

```
>PDF2Excel "E:\PDF Files\Table.pdf" -fods
```

to produce the same conversion results as the previous command.

Similarly, it is possible to convert the "Table.pdf" file to the legacy Microsoft Excel Format used before Office 2007. The argument passed on to the **-f** switch is the same as the extension of the legacy Excel Format: xls. The full command for this conversion is:

```
>PDF2Excel "E:\PDF Files\Table.pdf" -f"xls"
```

Finally, there is the Delimiter Separated Format, which is a simple plain text file format where the line breaks in the file represent separate table rows, and the delimiters (usually commas) represent separate column lines.

For this format you can again use any of the following commands for conversion of (in this example) a single file:

```
>PDF2Excel "E:\PDF Files\Table.pdf" -f"dsv"
```

**Note:** The extension of .dsv is not always recognized by the OS properly. You can either change the extension of converted files to .csv or force open the file with either MS Excel, Open Office Calc or a plain text reader, such as Notepad.

#### 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 (<startPageN°>), end page number (<endPageN°>) or both. When specifying only starting page, followed by a hyphen ( - ), the assumed end page is the end of the document. Similarly, when specifying only last page number, the assumed starting page is the first page of the document.

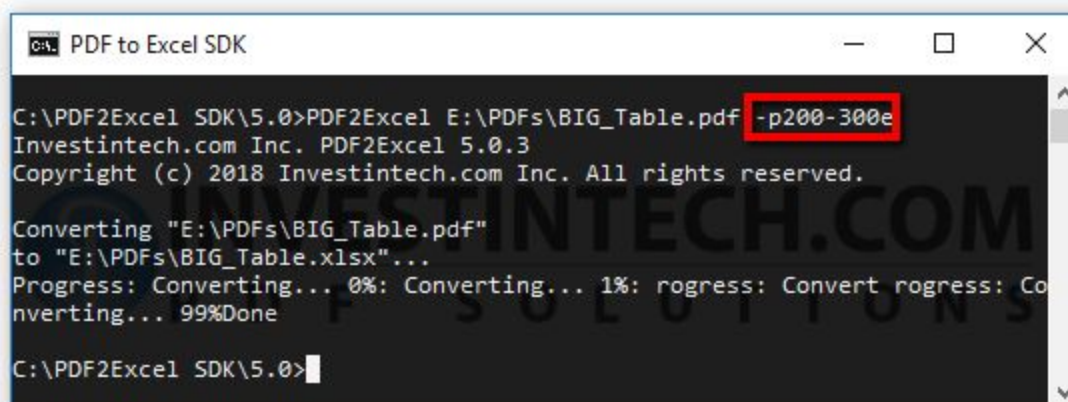
Additionally, you can decide to extract only odd or even pages of the document by using suffixes, **o** or **e** respectively. The table below provides you with an overview of the possible syntax for each case and what it does:

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 <startPageN°> 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 <startPageN°>.

**Example:** Let's say you have a large PDF file containing entries in tabular data, but you only require even pages from page 200 to page 300. You can use the following command to extract specific pages:

```
>PDF2Excel E:\PDFs\BIG_Table.pdf -p200-300e
```



```

C:\PDF2Excel SDK\5.0>PDF2Excel E:\PDFs\BIG_Table.pdf -p200-300e
Investintech.com Inc. PDF2Excel 5.0.3
Copyright (c) 2018 Investintech.com Inc. All rights reserved.

Converting "E:\PDFs\BIG_Table.pdf"
to "E:\PDFs\BIG_Table.xlsx"...
Progress: Converting... 0%: Converting... 1%: rogress: Convert rogress: Co
nverting... 99%Done

C:\PDF2Excel SDK\5.0>

```

#### 4.4.5. Using template files

The PDF to Excel automatic conversion is fine-tuned to provide good conversion results on large sets of different kinds of tables in different PDF documents.

While the Page Ranges function gives you some control over the content that is being converted, in some cases automatic conversions might not meet your requirements. There are two major reasons why this may happen:

- You have unusual or complex table structures in PDF file(s)
- You have a specific subset of data you would like extracted from the pages of PDF files

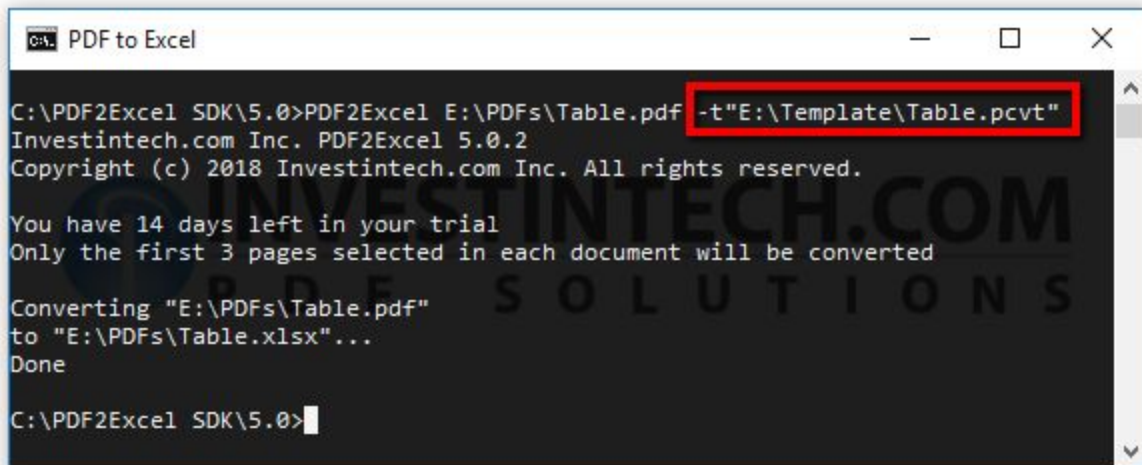
Whichever the case, the Command Line Tool, with the assistance of our Able2Extract desktop application provides a solution to this problem. For each of the files, you can specify a special Template file.

Since the creation of template files requires visual inspection of the table structure of the PDF document in question, Able2Extract is needed for creating those template files. To learn more on how to make template files, please refer to our online documentation and knowledge base on the Investintech website [www.investintech.com](http://www.investintech.com), or refer directly [to this link](#) for the help files in PDF format. Chapter 6: Conversion options for template Files also provides more information on this subject.

In order to use template files when converting files using CLT, you need to provide a relative or absolute path of the file to the `-t` switch.

Here is the example command for converting "Table.pdf" using "Table.pcv" template:

```
>PDF2Excel E:\PDFs\Table.pdf -t"E:\Template\Table.pcvt"
```



The output file should now contain exactly the data specified in the ".pcvt" template file.

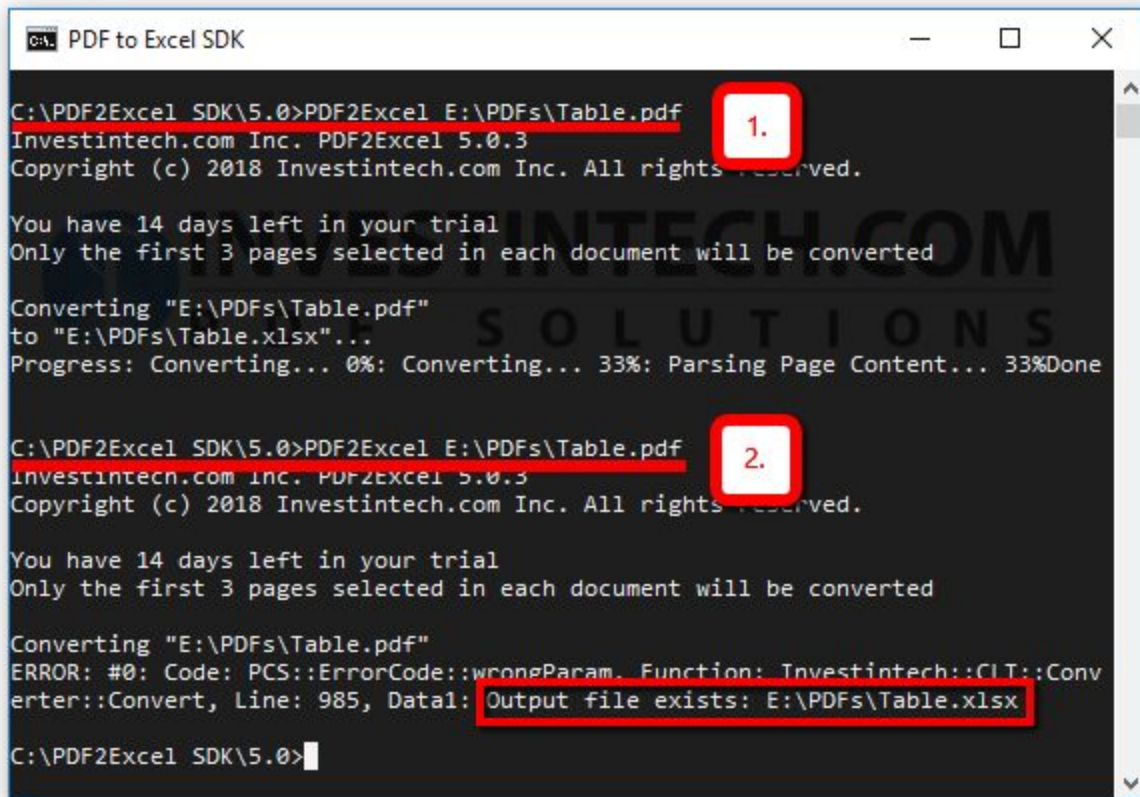
**Note:** PDF to Excel SDK version 5.0 works best with template files created with Able2Extract Desktop Application version 14.0. However, the template files created with older Able2Extract software versions are also supported, including the templates created with Able2Extract version 7.0, which have a different file extension: ".ta2e"

#### 4.4.6. 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\Table.pdf`, you will see the following error in the console:

```
Output file exists: E:\PDFs\Table.xlsx
```



```
C:\PDF2Excel SDK\5.0>PDF2Excel E:\PDFs\Table.pdf
Investintech.com Inc. PDF2Excel 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\Table.pdf"
to "E:\PDFs\Table.xlsx"...
Progress: Converting... 0%: Converting... 33%: Parsing Page Content... 33%Done

C:\PDF2Excel SDK\5.0>PDF2Excel E:\PDFs\Table.pdf
Investintech.com Inc. PDF2Excel 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\Table.pdf"
ERROR: #0: Code: PCS::ErrorCode::wrongParam. Function: Investintech::CLT::Converter::Convert, Line: 985, Data1: Output file exists: E:\PDFs\Table.xlsx

C:\PDF2Excel 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 will happen when collision occurs ?	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 4.4.7](#)).

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:

```
>PDF2Excel E:\PDFs\Table.pdf -e"o"
```

```
>PDF2Excel E:\PDFs\Table.pdf -e"m"
```

```
>PDF2Excel E:\PDFs\Table.pdf -e"s"
```

Of course, in the third case, no conversion is actually performed. The engine will simply inform you that the output file name already exists.

#### 4.4.7. Verbosity level of Console output

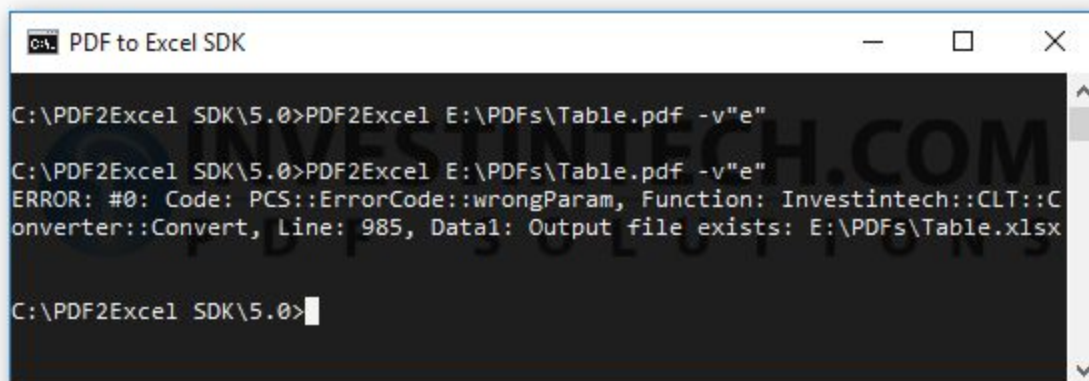
Using the Command Line Tool is often done in a fully automated manner. When this is the case, it is often unnecessary or even counterproductive to have all of the conversion outputs to be printed to the command line interface.

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 4.4.7](#), 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 "Table.pdf" is shown below:

```
>PDF2Excel E:\PDFs\Table.pdf -v"e"
```



```
C:\PDF2Excel SDK\5.0>PDF2Excel E:\PDFs\Table.pdf -v"e"
C:\PDF2Excel SDK\5.0>PDF2Excel E:\PDFs\Table.pdf -v"e"
ERROR: #0: Code: PCS::ErrorCode::wrongParam, Function: Investintech::CLT::C
onverter::Convert, Line: 985, Data1: Output file exists: E:\PDFs\Table.xlsx
C:\PDF2Excel 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.

#### 4.4.8. 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 parameter to the **-w** switch.

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

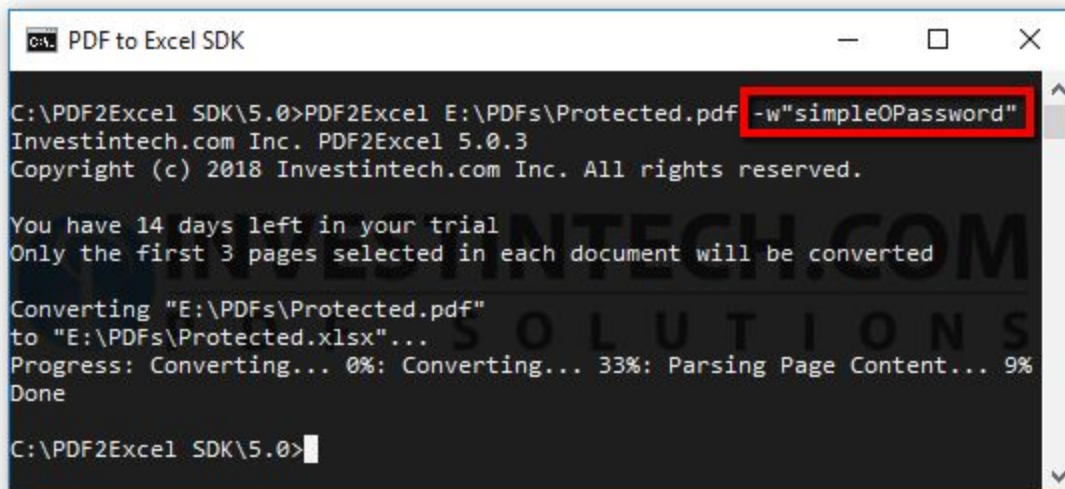


**ERROR: #0: Code: PDL::ERR\_PDF\_WRONG\_PASSWORD**

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 "simpleOPassword":

```
>PDF2Excel E:\PDFs\Protected.pdf -w"simpleOPassword"
```



```
C:\PDF2Excel SDK\5.0>PDF2Excel E:\PDFs\Protected.pdf -w"simpleOPassword"
Investintech.com Inc. PDF2Excel 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\Protected.pdf"
to "E:\PDFs\Protected.xlsx"...
Progress: Converting... 0%: Converting... 33%: Parsing Page Content... 9%
Done

C:\PDF2Excel SDK\5.0>
```

#### 4.4.9. 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 PDF to Excel 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.

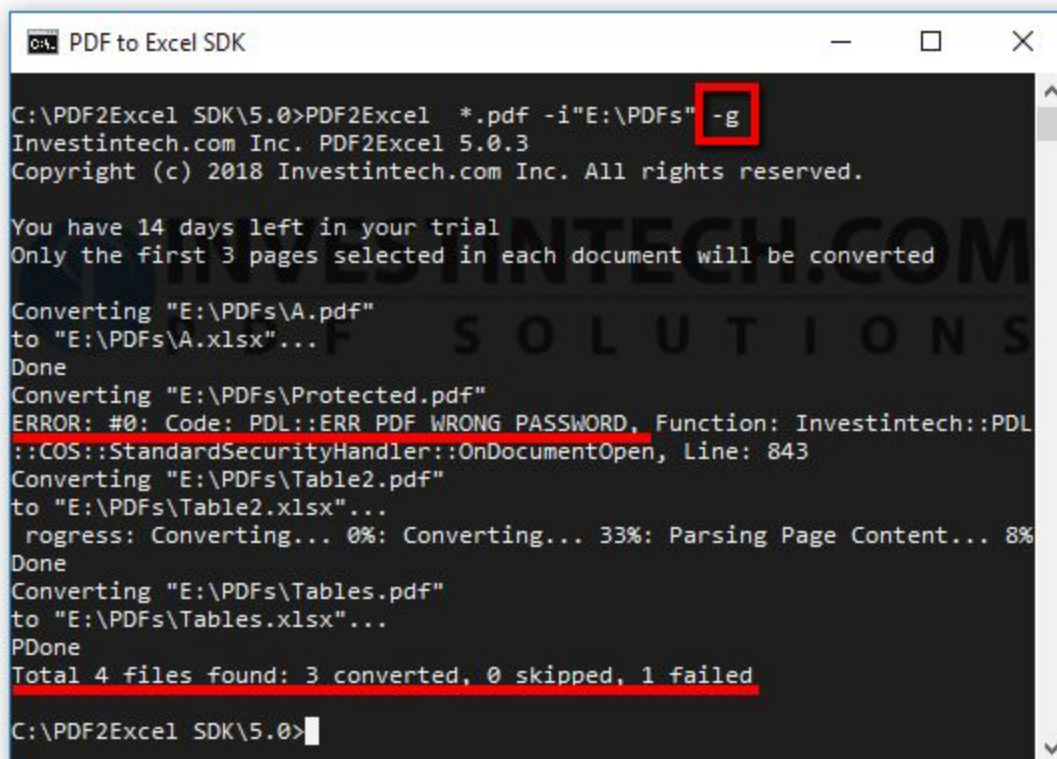


**Example:** You have a directory named "E:\PDFs\" with several .pdf files, one of which is password protected. You can order the conversion of all .pdf files in the directory with the following command:

```
>PDF2Excel *.pdf -i"E:\PDFs"
```

If the 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:

```
>PDF2Excel *.pdf -i"E:\PDFs" -g
```



```
C:\PDF2Excel SDK\5.0>PDF2Excel *.pdf -i"E:\PDFs" -g
Investintech.com Inc. PDF2Excel 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\A.pdf"
to "E:\PDFs\A.xlsx"...
Done
Converting "E:\PDFs\Protected.pdf"
ERROR: #0: Code: PDL::ERR PDF WRONG PASSWORD, Function: Investintech::PDL
::COS::StandardSecurityHandler::OnDocumentOpen, Line: 843
Converting "E:\PDFs\Table2.pdf"
to "E:\PDFs\Table2.xlsx"...
Progress: Converting... 0%: Converting... 33%: Parsing Page Content... 8%
Done
Converting "E:\PDFs\Tables.pdf"
to "E:\PDFs\Tables.xlsx"...
Done
Total 4 files found: 3 converted, 0 skipped, 1 failed
C:\PDF2Excel SDK\5.0>
```

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

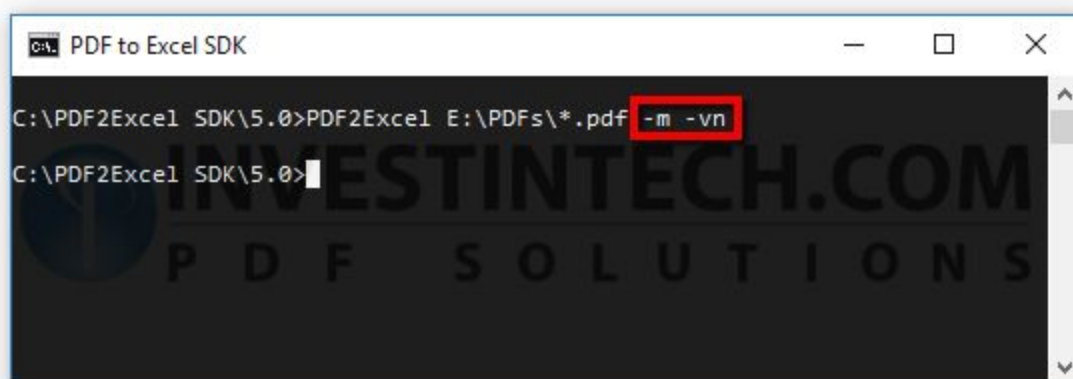
#### 4.4.10. Parallel Conversion

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:

```
PDF2Excel E:\PDFs\*.pdf -m -vn
```



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

## 5. Using the Shared Library

The PDF to Excel SDK comes with a complete set of files which allow for quick integration into your development environment. PDF2Excel.dll is a collection of methods compiled, linked and stored in a Dynamic Link Library. You can use the Dynamic Link Library PDF2Excel.dll directly, or through one of the proxy libraries for COM and .NET for non C/C++ projects.

### 5.1. Preparation

After installing the PDF to Excel SDK tool on your local machine you should have access to Sample Files for Visual studio 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\PDF2Excel 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.

These instructions will assume you have Visual Studio 2017 installed on the machine you intend to use for integrating the PDF2Excel Shared Library

#### 5.1.1. Licence Registration

In order to use a fully licenced version of PDF2Excel.dll in your application, you will need two distinct components:

1. Personalised Licence.lic file
2. 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 you must pass an argument to the initialization function in PDF2Excel class (C++):

```
PCSErrorCode PDF2ExcelInitialize( const char* 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.

**Note:** Return type of `PDF2ExcelInitialize()` is `PCSErrorCode`. This is an alias of type integer (int) which is defined in the `Lib.h` file.

## 5.2. Lib.c sample project

This section will explain how to connect the Visual C++ sample project with `PDF2Excel.dll` library.

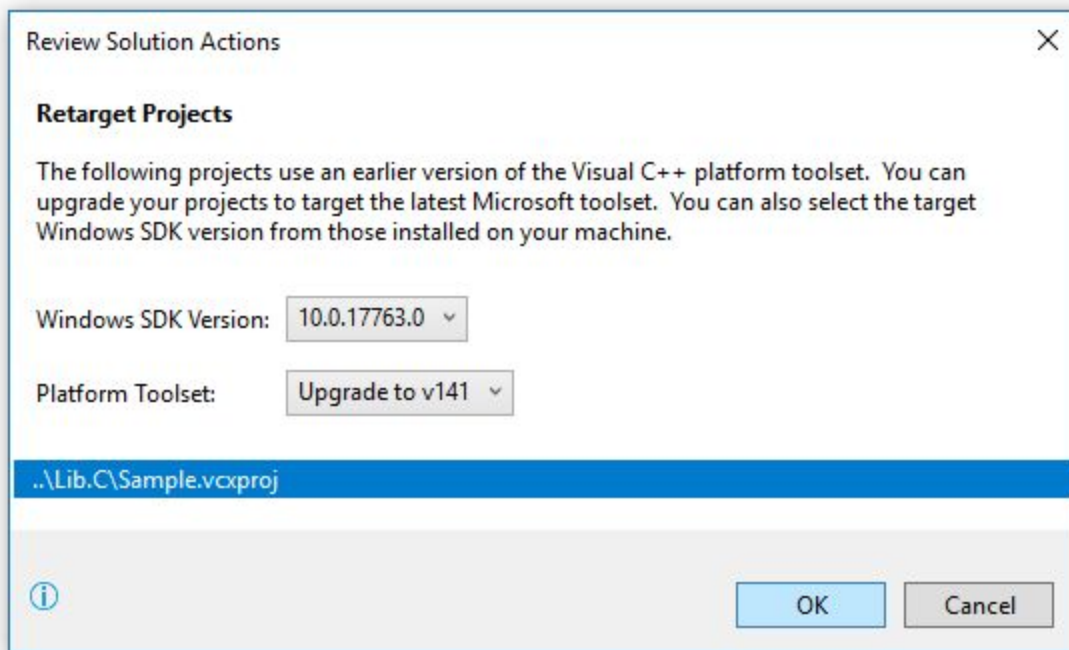
This sample project is placed by default in the `%USERPROFILE%\Documents\PDF2Excel\Samples\Lib.C` directory.

**Important:** To make the process of resolving dependencies easier, it is best to copy all the files from the PDF2Excel SDK install directory to the `Lib.C` directory.

Once you have copied the required files, open `Sample.sln` either by directly clicking on the file itself or by opening the solution from Visual Studio by using `File -> Open/Project Solution`.

During the opening of the Solution, you will most likely be asked to retarget the Visual C++ platform toolset. It is recommended to accept this project upgrade and leave the default options recommended by Visual Studio.

The following screenshot displays this type of window for retargeting Projects.



**Note:** The actual Windows SDK Version and platform Toolset may depend on your local environment, and thus may differ somewhat from the screenshot above.

If the upgrade is successful, you should see several messages in the Output Window, one for every build configuration. The messages should be formatted in a manner similar to the following:

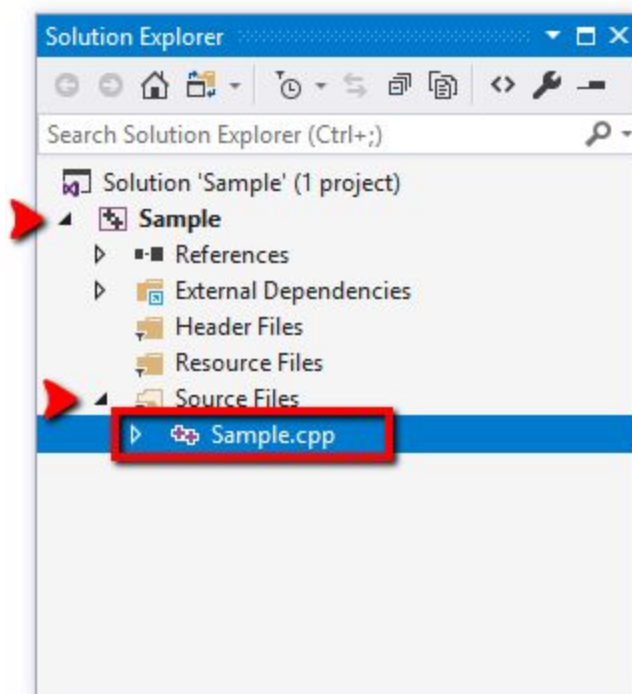
**changing Platform Toolset to 'v141' (was 'v100')**

If the Output Window is not visible you can open it from the Menu: View -> Output.

Visual Studio will not initially open any files. To open the source file of a Sample Project for editing in Visual Studio, look for the Solution Explorer, by default it is located on the right hand side. If the Solution Explorer window is not visible, you can open it from the Toolbar, by choosing View -> Solution explorer. Alternatively you can use the keyboard shortcut, CTRL + W, S.

Next, you might need to expand the "Sample" project and Sample files, if not already expanded. Then open Sample.cpp file in the editor by double-clicking, or using the right-click context menu and choosing Open.

You can refer to the following screenshot for quick reference:



### 5.2.1. Resolving Dependencies

Even if all of the necessary files from the PDF2Excel SDK are present in the same directory as Sample.sln file, some dependency issues might occur.

If the header file PDF2Excel.h is not resolved when the solution is opened, you might need to make a simple correction on how the preprocessor searches for the header. This file is included in the sample project in the following way:

```
#include <PDF2Excel.h>
```

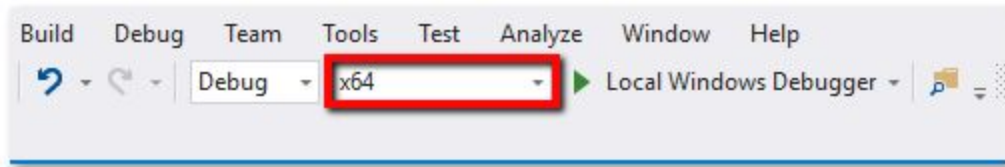
Simply changing the square brackets to parentheses will force the preprocessor to first search in the folder where the source file is located. Thus, you can replace the above line with the following:

```
#include "PDF2Excel.h"
```

If this relative path cannot be resolved, it might be necessary to provide the absolute path of the PDF2Excel header file. In order to do this, navigate and note down the relative location of this header file. Add the absolute path instead of the relative one:

```
#include <C:\PDF2Excel SDK\5.0\PDF2Excel.h>
```

Before building the project, you should check if the correct CPU architecture is selected in Configuration manager. This option is usually quickly accessible directly from the toolbar, left of the "Run" button, as presented in the following screenshot.



Use the configuration that matches the target platform of the PDF2Excel SDK tools.

### 5.2.2. Running the sample project

When all of the dependencies are resolved, and you can successfully build the Solution, running the Solution without any parameters set will fail with exit code 1.

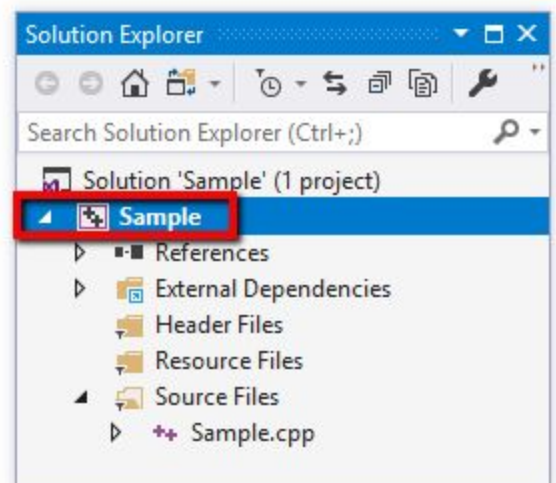
The out-of-the-box Sample.cpp has a check for the number of parameters passed as command line argument. For the conversion to be successful, you need to pass at least two arguments to the application: Input file path (PDF) and output file path.

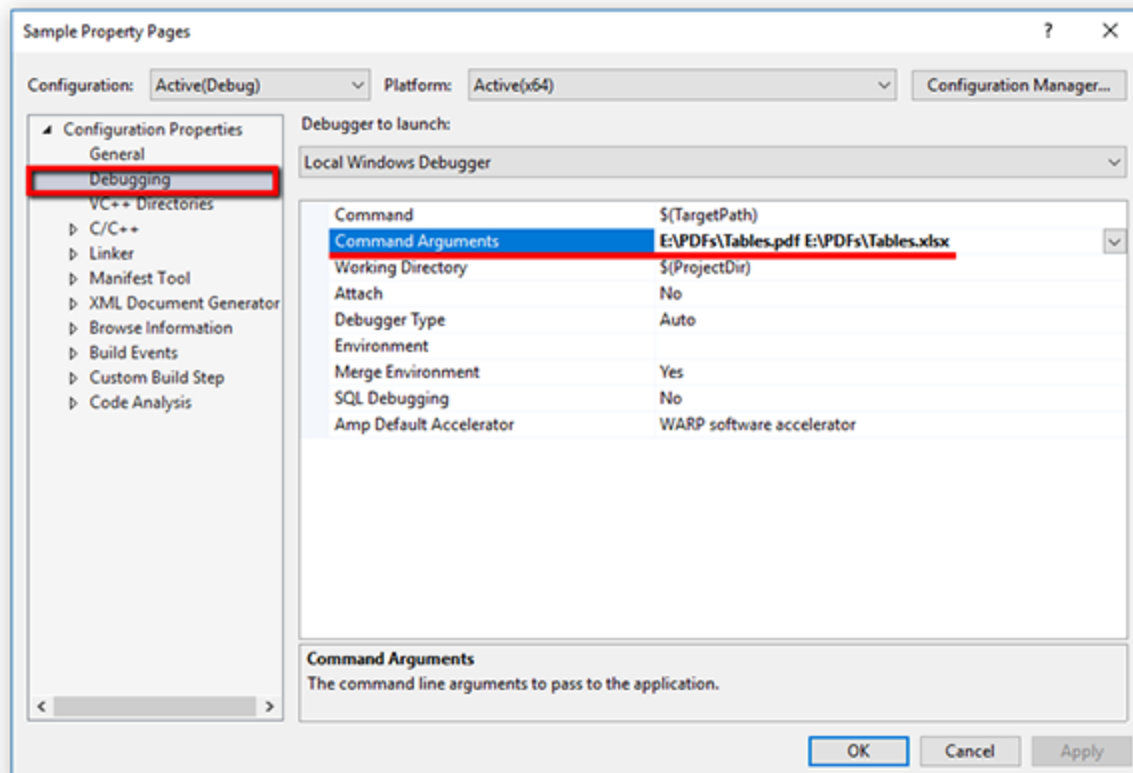
To pass on parameters you can use one of the following two approaches:

1. Directly run the `...\Debug\Sample.exe` executable from Command Line, which Visual Studio built for you, and pass the required parameters separated by "white space".

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, as marked in the screenshot.

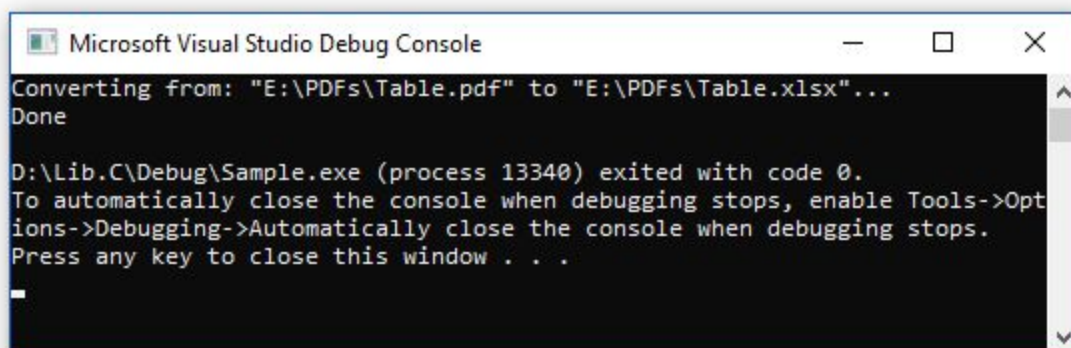
From the Sample Project Property dialog, on the left hand side, choose "Debugging" and fill in two paths for the Command Arguments field on the right. The first Argument is the path to the Input File that should be converted, while the second argument is the path for the output file, separated by "white space".





**Note:** The default output format for the conversion is Microsoft Office 2007 file. In order for the output file to be properly interpreted by the Office application, it is strongly recommended for the output file extension to be .xlsx. This nomenclature can be seen in the screenshots. 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. The default output format of the output file is Microsoft Office 2007 file, regardless of file extension given as an argument. To change the actual output file format, refer to Section 5.2.4.

### 5.2.3. Registration

PDF2Excel.dll will convert only the first three pages of the document when running in trial mode. You can acquire the licence from the official Investintech website:

<https://www.investintech.com>

For successful and unlimited conversions, you will be provided with two components: a Licence.lic file and a personal password which should be passed on as an argument to the method:

```
PCSErrorCode PDF2ExcelInitialize( const char* pass)
```

The Licence.lic file should be placed in the same directory as the Sample.sln file. To initialize PDF2Excel.dll, you need to pass password String to the function `PDF2ExcelInitialize()`, replacing the NULL placeholder. You can use the following code snippet from Sample.cpp as a reference:

```
if((errorCode=PDF2ExcelInitialize( NULL ))!=IT_PCS_ERR_SUCCESS )
{
    wcout<< L"Initialization failed: "<< errorCode << endl;
    return errorCode;
}
```

As you can see, this snippet, included in the Sample file, provides basic error checking, giving the user information if the Licenced Initialization has failed.

### 5.2.4. Choosing different output format

There are four acceptable values for the output conversion. To change the output format, you need to change the third argument of the function `PDF2Excel()`, which accepts the type of unsigned char.

Available macro values can be accessed in the header file `PDF2Excel.h`. You can also refer to the following table for the names, raw and unsigned char values as well as a short description of the different output formats.

Variable name	Output format	Value
IT_PDF2EXCEL_FORMAT_DSV	Comma separated value ( plain text )	1
IT_PDF2EXCEL_FORMAT_MSEXCEL21	Pre 2007 MS Excel (MS Excel 2.1)	2
IT_PDF2EXCEL_FORMAT_MSEXCEL2007	MS Excel format	3
IT_PDF2EXCEL_FORMAT_OOCALC	Open Office Document spreadsheet	4

**Example:** To change the output format to Open Office Document spreadsheet you would need to change your call to the `PDF2Excel()` function to match the following code snippet.

```
if( (errorCode = ::PDF2Excel(  argv[ 1 ],
                             argv[ 2 ],
                             IT_PDF2EXCEL_FORMAT_OOCALC,
                             argc == 3 ? NULL : argv[ 3 ],
                             NULL )) != IT_PCS_ERR_SUCCESS )
{
    std::wcout << L"Conversion failed: " << errorCode << std::endl;
    return errorCode;
}
```

While possible, it is not recommended to use raw values instead of a macro.

Note that the output file extension doesn't change automatically, and it is only determined trivially as a string (C-style string) value. Thus, when changing your output file format value, the developer should also provide an appropriate output file extension.

### 5.3. Net C# sample project

This section is intended to simplify connecting the `PDF2Excel.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\PDF2Excel 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 PDF2Excel.dll is running the "Sample.sln" file located in the ".\Net.C#" directory. This can be accomplished either by using Open Project/Solution from 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 Sample.sln file.

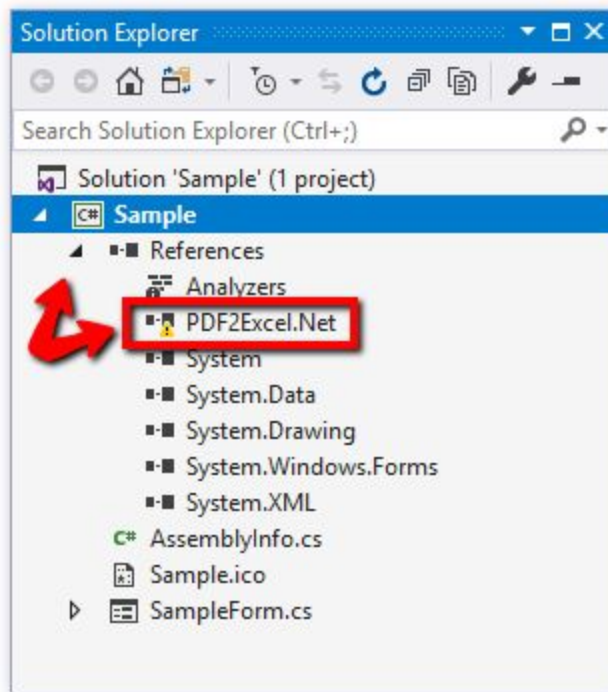
The directories that are of interest here are the sub-directories of the "bin" directory, which will store executables after the first build of the project. The name of the directory will depend on the Build Configuration used to build the application. Since the default configuration is "Debug" we will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

Next, using Windows Explorer, navigate to the installation directory of PDF to Excel 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 PDF2Excel.Net.dll dependencies has 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.

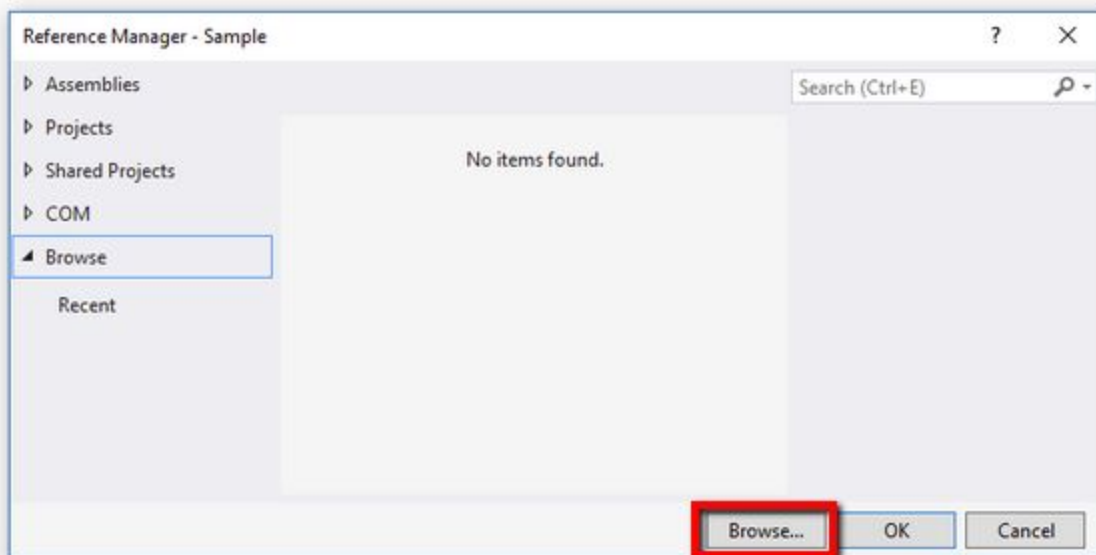
Solution Explorer should look similar to this:



If marked by a warning exclamation mark, PDF2Excel.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 PDF to Excel SDK tools. This is usually the installation directory chosen during the installation. Once you have navigated to the location of PDF to Excel SDK tools, choose and add PDF2Excel.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()
{
    PDF2Excel.Initialize("XXXXXXX");
    Application.Run(new SampleForm());
    PDF2Excel.UnInitialize();
}
```

Argument of the method `public static void Initialize(string pass)` needs to be changed in order to use PDF2Excel 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.

First find the following line:

```
PDF2Excel.Format format = PDF2Excel.Format.CSV;
```

And change it with:

```
PDF2Excel.Format format = PDF2Excel.Format.DSV;
```

This command simply declares and initialises enum used to determine the desired output format of the conversion file. The actual format is chosen based on user choice from the drop down menu (formatComboBox) of the form.

```
switch (formatComboBox.SelectedIndex)
{
    case 0: format = PDF2Excel.Format.DSV; break;
    case 1: format = PDF2Excel.Format.MSExcel21; break;
    case 2: format = PDF2Excel.Format.MSExcel2007; break;
    case 3: format = PDF2Excel.Format.OOCalc; break;
}
```

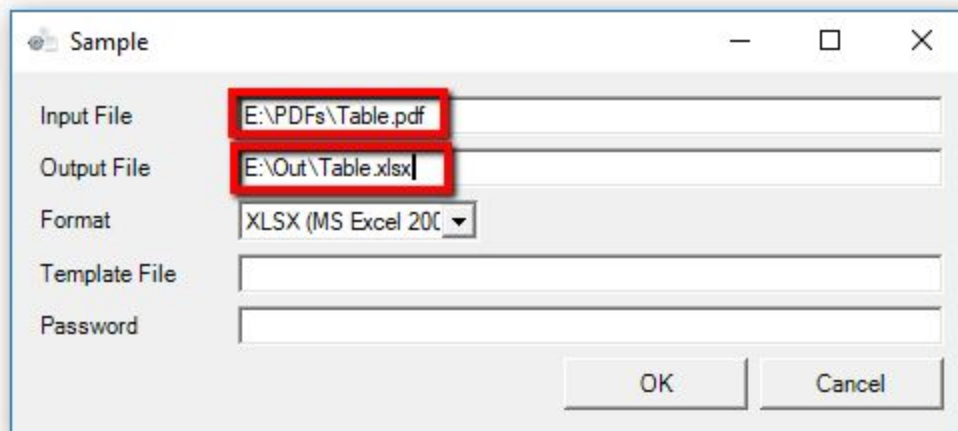
There are four possible output file formats. These are outlined in the following table:

Enum Value	Compatible with	Expected extension
Format.DSV	Comma separated value ( plain text )	csv
Format.MSExcel21	Pre 2007 MS Excel format (MS Excel 2.1)	xls
Format.MSExcel2007	MS Excel format	xlsx
Format.OOCalc	Open/Libre Office Document spreadsheet	ods

**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.

### 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:



The last step needed before the PDF2Excel 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 PDFtoExcel.dll

**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.

Optionally, you can choose different output formats for converted files, or use Template Files created with Able2Extract. You can read more about Template Files in Section 5.

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.



## 6. Conversion options for template Files

PDF to Excel tools provide you with several different options when converting files. However, the options provided by switches for the Command Line Tool and additional function arguments provided by the Shared Library are not the only way to take control of the conversion output.

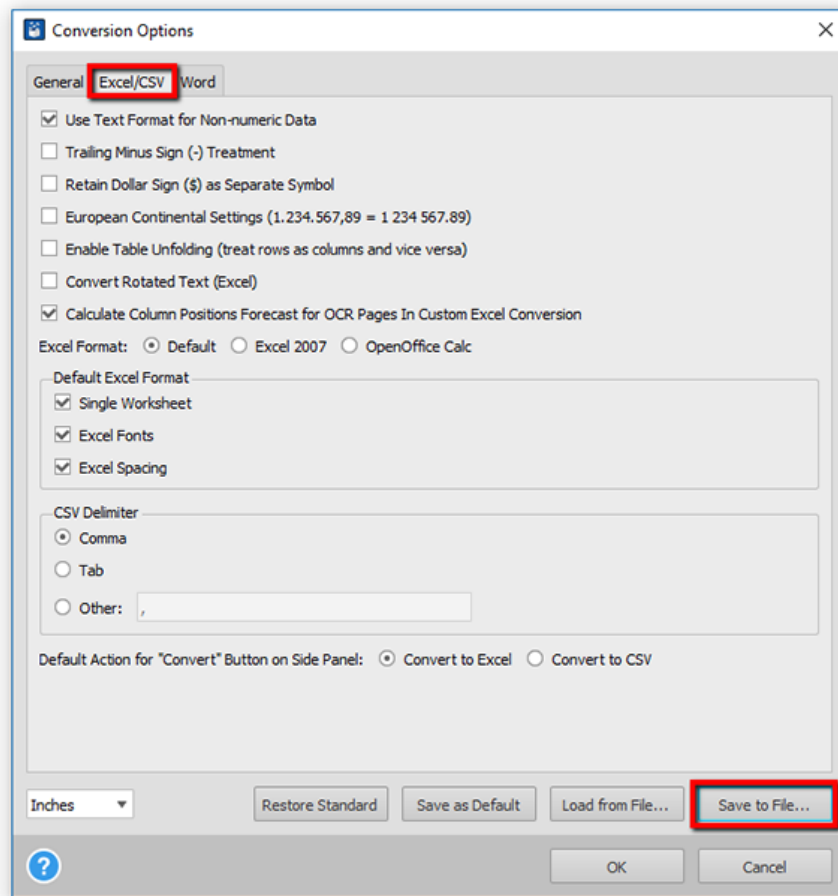
There are several more options that can be passed on to the conversion engine to tailor the conversion output to your specific needs. This chapter will give you insight on how to tap into this powerful feature for customization of PDF to Excel SDK tools.

### 6.1. Command Line Tool: Options.pcvr

When using the Command Line tool without the `-t` switch ([Section 4.4.5](#)) to specify a template file, a special Options.pcvr file is loaded. This file can be used to hold additional options for the conversions, such as conversion to a multi-worksheet Excel file, the treatment of trailing - signs, using European numbers standards for thousand and decimal delimiters, and several more.

This file is placed in the installation directory, which means that it usually resides in the same directory as the PDF2Excel.exe Console Application. Similar to template files which provide control of what data to extract, and how to place them in proper columns, this file provides control over the output document.

The best way to make changes to this file is by using the Able2Extract application to select the desired conversion options and export them to a file. To access this menu, open the Able2Extract Desktop application, go to View -> Conversion Options from the Menu Bar and click on the Excel/CSV tab. You will get the following screen:



You can modify any of the options in this dialog, then save it as an Options.pcvf file. After this, you can copy this options file to the installation directory replacing the default one.

**Caution:** It is advisable to either rename the original Options.pcvf file or copy the file to some other location as a backup. If the file is invalid you will get the following error:

**SAXParseException: Tag mismatch in line 9 column 4**

If you accidentally lose the original Options file, you can simply recreate it using the Able2Extract Desktop application by selecting the "Restore Standard" button from the Conversion Options dialog.

The specifics of each of the options in the Conversion Options dialog are beyond the scope of this guide. For further insight on each of the options, please refer to our in-app or online help files from the [www.investintech.com](http://www.investintech.com) website.

## 6.2. Shared Library Options.pcvt

Including additional options using functions from the shared library is also available. The primary `PDF2Excel.Convert()` function accepts .pcvt file as fourth argument.

Besides its main purpose, which is to load the Template file for conversion, this argument can also be used to load a Conversion Options pcv file. To check how this works, you can use the Sample files to try to load your Conversion Options file in place of a template, or provide your own implementation. For more information on creating a template file please refer to [Section 6.1](#).

## 6.3. Combining template file and options file

In [Section 6.1](#), it is mentioned that loading a template file which contains a saved table structure, for a specific document (or a set of documents) will cause the engine to ignore an Options.pcv file. In the case of a shared library, this is even more apparent, since either the Conversion Options or the Template file can be passed as an argument to `Convert()` function.

Inclusion of additional conversion options inside your template file can be easy. Before using Able2Extract Desktop app to create a conversion template file, simply open Conversion Options and either modify options to your specific need, or use the "Load from file..." button to load previously saved conversion options.

When you're satisfied with the table layout from your Custom Excel panel, saving the template file will automatically save your Conversion options, as well.

**Note:** If you already have a template file, you can quickly add different Conversion Options by modifying them first before using File -> Load Custom Excel Template to load them. After loading the file, you can save it under a different name.

## 7. Appendices

### 7.1. 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: <file Path/Name>	<ol style="list-style-type: none"><li>1. Use absolute file path example: "E:\PDFs\SomeFile.pdf"</li><li>2. Check the spelling</li><li>3. Use wildcard characters and/or recursion switch <b>-r</b> (section 3.4.1)</li></ol>
Output file exists: <file Path/Name>	<ol style="list-style-type: none"><li>1. Specify a different output Directory (section 3.4.2)</li><li>2. Delete the previously created file at the location &lt;file Path&gt;</li><li>3. Use <b>-eo</b> or <b>-em</b> to overwrite or modify your output file name (section 3.4.6)</li></ol>
Unrecognized input file format: <file Path/Name>	<ol style="list-style-type: none"><li>1. When using multiple file conversions with wildcards:<ol style="list-style-type: none"><li>1.1. Increase restrictions on mask Example: use .pdf extension (*.pdf)</li><li>1.2. Use <b>-g</b> switch to skip error (section 3.4.9)</li></ol></li><li>2. When using single file conversion:<ol style="list-style-type: none"><li>2.1. File format is not supported.</li></ol></li></ol>

Problem	Possible Solutions
SAXParseException: Tag mismatch in line 9 column 4	<ol style="list-style-type: none"> <li>1. Create or choose another template file (.pcvt)</li> <li>2. Re-create an Option.pcv file in Able2Extract Desktop App (Section 6.1)</li> </ol>
Initialization failed: 10	<ol style="list-style-type: none"> <li>1. Check for password and Licence integrity</li> </ol>
PCS::ErrorCode::wrongParam	<ol style="list-style-type: none"> <li>1. Check if correct parameters are used. For reference see (Section 3.4)</li> <li>2. Make sure no space is provided between a flag and the passed parameter.</li> </ol>
The filename, directory name, or volume label syntax is incorrect.	<ol style="list-style-type: none"> <li>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.</li> </ol>

## 7.2. 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 'PDF2Excel.dll'	<ol style="list-style-type: none"><li>1. Make sure that the required files and directories are in the same directory as binaries (see section 5.2.1)</li></ol>
Warning LNK4272 library machine type 'x64' conflicts with target machine type 'x86'	<ol style="list-style-type: none"><li>1. Change your specified CPU platform architecture build from Configuration Manager in VS Section 5.2.1</li><li>2. Download the appropriate SDK build</li></ol>
Conversion failed: 3024 (File Save error)	<ol style="list-style-type: none"><li>1. Change your output file name</li><li>2. Delete the previous output file</li></ol>
Conversion failed: 5 Unrecognized File Type	<ol style="list-style-type: none"><li>1. Check the input file name for spelling errors</li><li>2. Include proper file extensions of the input file name</li><li>3. Choose .pdf file for input file</li></ol>
Conversion failed: 3022 File Create	<ol style="list-style-type: none"><li>1. Change output file name</li><li>2. Check for Resource folder naming collision, do one of the following: delete, rename or move resource folder in the output dir.</li></ol>