

**Investintech.com Inc.  
Software Development Kit:  
PDF-to-Excel Function Library  
User's Guide**

**May 25, 2007**  
**<http://www.investintech.com>**

Copyright 2007 Investintech.com, Inc. All rights reserved

Adobe® is registered by Adobe Systems Incorporated  
Acrobat® is registered by Adobe Systems Incorporated  
Postscript® is registered by Adobe Systems Incorporated  
Access™ is registered by Microsoft Corporation  
Excel® is registered by Microsoft Corporation  
Visual Basic® is registered by Microsoft Corporation  
Visual C++® is registered by Microsoft Corporation  
Visual C#® is registered by Microsoft Corporation  
Visual J#® is registered by Microsoft Corporation  
Visual Studio® is registered by Microsoft Corporation  
Win32® is registered by Microsoft Corporation  
Windows® is registered by Microsoft Corporation  
Windows NT® is registered by Microsoft Corporation  
Windows Vista™ is registered by Microsoft Corporation



# Contents

<b>Preface .....</b>	<b>1</b>
About This Documentation .....	1
Typographical Conventions Used in This Document .....	1
Getting More Information .....	1
Customer Service and Technical Support .....	2
Fax and Mailing Address.....	2
<b>Investintech PDF2Excel Conversion DLL .....</b>	<b>3</b>
What is the Investintech PDF2Excel Conversion DLL?.....	4
Installation Instructions for PDF2ExcelDLL .....	4
System Requirements .....	4
Starting the Installation.....	4
Using the Investintech PDF2Excel Conversion DLL in Microsoft Visual C++ .NET 2003 .....	11
Implicit Linking.....	11
Linking .lib file with project .....	12
Using methods from DLL file.....	13
Investintech PDF2Excel Conversion DLL Methods.....	13
Interface .....	13
Parameter Type.....	14
File Names .....	14
Error Handling.....	14
Common Sample Source Code.....	15
Conversion from PDF document to Excel Workbook.....	15
PDF_to_Excel_CBR.....	15
Prototype .....	15
Description .....	15
Calling Convention .....	15
Parameters .....	16
Returns .....	16
Example .....	16
PDF_to_Excel_CBT.....	16
Prototype .....	16
Description .....	16
Calling Convention .....	16
Parameters.....	16
Returns .....	16
Example .....	16
PDF_to_Excel_CLR.....	17
Prototype .....	17
Description .....	17
Calling Convention .....	17
Parameters.....	17
Returns .....	17
Example .....	17
PDF_to_Excel_CLT .....	17
Prototype .....	17
Description .....	18

Calling Convention.....	18
Parameters.....	18
Returns.....	18
Example.....	18
PDF_to_Excel_SBR.....	18
Prototype.....	18
Description.....	18
Calling Convention.....	18
Parameters.....	19
Returns.....	19
Example.....	19
PDF_to_Excel_SBT.....	19
Prototype.....	19
Description.....	19
Calling Convention.....	19
Parameters.....	19
Returns.....	19
Example.....	19
PDF_to_Excel_SLR.....	20
Prototype.....	20
Description.....	20
Calling Convention.....	20
Parameters.....	20
Returns.....	20
Example.....	20
PDF_to_Excel_SLT.....	20
Prototype.....	20
Description.....	21
Calling Convention.....	21
Parameters.....	21
Returns.....	21
Example.....	21
Conversion from PDF document to Excel Workbook using a template file.....	21
PDF_to_Excel_Template_CBR.....	21
Prototype.....	21
Description.....	22
Calling Convention.....	22
Parameters.....	22
Returns.....	22
Example.....	22
PDF_to_Excel_Template_CBT.....	22
Prototype.....	22
Description.....	22
Calling Convention.....	22
Parameters.....	23
Returns.....	23
Example.....	23
PDF_to_Excel_Template_CLR.....	23
Prototype.....	23
Description.....	23
Calling Convention.....	23
Parameters.....	24

Returns .....	24
Example .....	24
PDF_to_Excel_Template_CLT .....	24
Prototype .....	24
Description .....	24
Calling Convention .....	24
Parameters .....	24
Returns .....	25
Example .....	25
PDF_to_Excel_Template_SBR .....	25
Prototype .....	25
Description .....	25
Calling Convention .....	25
Parameters .....	25
Returns .....	26
Example .....	26
PDF_to_Excel_Template_SBT .....	26
Prototype .....	26
Description .....	26
Calling Convention .....	26
Parameters .....	26
Returns .....	26
Example .....	26
PDF_to_Excel_Template_SLR .....	27
Prototype .....	27
Description .....	27
Calling Convention .....	27
Parameters .....	27
Returns .....	27
Example .....	27
PDF_to_Excel_Template_SLT .....	28
Prototype .....	28
Description .....	28
Calling Convention .....	28
Parameters .....	28
Returns .....	28
Example .....	28
Convert from PDF document to CSV file .....	29
PDF_to_CSV_CBR .....	29
Prototype .....	29
Description .....	29
Calling Convention .....	29
Parameters .....	29
Returns .....	29
Example .....	29
PDF_to_CSV_CBT .....	29
Prototype .....	29
Description .....	29
Calling Convention .....	30
Parameters .....	30
Returns .....	30
Example .....	30

PDF_to_CSV_CLR.....	30
Prototype.....	30
Description.....	30
Calling Convention.....	30
Parameters.....	31
Returns.....	31
Example.....	31
PDF_to_CSV_CLT.....	31
Prototype.....	31
Description.....	31
Calling Convention.....	31
Parameters.....	31
Returns.....	31
Example.....	31
PDF_to_CSV_SBR.....	32
Prototype.....	32
Description.....	32
Calling Convention.....	32
Parameters.....	32
Returns.....	32
Example.....	32
PDF_to_CSV_SBT.....	32
Prototype.....	32
Description.....	33
Calling Convention.....	33
Parameters.....	33
Returns.....	33
Example.....	33
PDF_to_CSV_SLR.....	33
Prototype.....	33
Description.....	33
Calling Convention.....	34
Parameters.....	34
Returns.....	34
Example.....	34
PDF_to_CSV_SLT.....	34
Prototype.....	34
Description.....	34
Calling Convention.....	34
Parameters.....	34
Returns.....	34
Example.....	35
Convert from PDF document to CSV file using template file.....	35
PDF_to_CSV_Template_CBR.....	35
Prototype.....	35
Description.....	35
Calling Convention.....	35
Parameters.....	35
Returns.....	36
Example.....	36
PDF_to_CSV_Template_CBT.....	36
Prototype.....	36

Description .....	36
Calling Convention .....	36
Parameters .....	36
Returns .....	36
Example .....	36
PDF_to_CSV_Template_CLR .....	37
Prototype .....	37
Description .....	37
Calling Convention .....	37
Parameters .....	37
Returns .....	37
Example .....	37
PDF_to_CSV_Template_CLT .....	37
Prototype .....	37
Description .....	38
Calling Convention .....	38
Parameters .....	38
Returns .....	38
Example .....	38
PDF_to_CSV_Template_SBR .....	38
Prototype .....	38
Description .....	39
Calling Convention .....	39
Parameters .....	39
Returns .....	39
Example .....	39
PDF_to_CSV_Template_SBT .....	39
Prototype .....	39
Description .....	39
Calling Convention .....	39
Parameters .....	40
Returns .....	40
Example .....	40
PDF_to_CSV_Template_SLR .....	40
Prototype .....	40
Description .....	40
Calling Convention .....	40
Parameters .....	41
Returns .....	41
Example .....	41
PDF_to_CSV_Template_SLT .....	41
Prototype .....	41
Description .....	41
Calling Convention .....	41
Parameters .....	41
Returns .....	42
Example .....	42
VB6 callable code .....	42
PDF_to_Excel_VB6 .....	42
Prototype .....	42
Description .....	42
Calling Convention .....	42



Parameters.....	42
Returns .....	43
Example .....	43
PDF_to_Excel_Template_VB6 .....	43
Prototype.....	43
Description.....	43
Calling Convention.....	43
Parameters.....	43
Returns .....	43
Example .....	43
Prototype.....	44
Description.....	44
Calling Convention.....	44
Parameters.....	44
Returns .....	44
Example .....	44
PDF_to_CSV_Template_VB6.....	44
Prototype.....	44
Description.....	45
Calling Convention.....	45
Parameters.....	45
Returns .....	45
Example .....	45
InvestintechConversionDLL_IDString.....	45
Description.....	45
Parameters.....	45
Returns .....	45
Investintech PDF-To-Excel Conversion COM Component Methods .....	46
PDF2Excel .....	46
Prototype.....	46
Description.....	46
Calling Convention.....	46
Parameters.....	46
Returns .....	46
Example.....	46
PDF_to_CSV_Template_VB6.....	47
Prototype.....	47
Description.....	47
Calling Convention.....	47
Parameters.....	47
Returns .....	47
Example .....	47
PDF_to_CSV_Template_VB6.....	48
Prototype.....	48
Description.....	48
Calling Convention.....	48
Parameters.....	48
Returns .....	48
Example .....	48
PDF_to_CSV_Template_VB6.....	49
Prototype.....	49
Description.....	49

Calling Convention .....	49
Parameters .....	49
Returns .....	49
Example .....	49
<b>Index .....</b>	<b>50</b>

## Preface

Welcome to Investintech.com Inc. Software Development Kit (SDK). This SDK provides technologies for converting PDF files into other file. This user's guide provides the basic information needed to use all parts of the SDK. This SDK consists of:

- Investintech PDF To Excel Conversion DLL, a Dynamic-Link Library for converting PDF files

This document provides a comprehensive description of the DLL.

## About This Documentation

This documentation consists of:

- Investintech PDF To Excel Conversion DLL, describing usage and methods of DLL designed to convert PDF files into various formats (Excel, CSV).

## Typographical Conventions Used in This Document

The following table describes typographical and naming conventions used in this document:

<i>Font</i>	<i>Used for</i>	<i>Examples</i>
monospaced	Expected input from the user	a2ecl -pdf2text abc.pdf abc.txt
	Paths and filenames	c:\a2ecl\a2ecl.exe
	Source code	MessageBox("ABC");
monospaced blue	Source code keywords	If
monospaced green	Source code comments	//this is a comment
monospaced bold	Method name	The <b>AboutBox()</b> method
'Single quoted bold'	Elements of the user interface like buttons, icons etc.	click on ' <b>Start</b> ' button, press ' <b>Enter</b> '
blue underlined	Internet links	<a href="http://www.investintech.com">http://www.investintech.com</a>

## ***Getting More Information***

Investintech.com Inc. conducts research in data conversion technology and develops data conversion products utilizing results of the research. More information about our company is available at the company's website <http://www.investintech.com>.

## **Customer Service and Technical Support**

Investintech.com Inc. strives to provide the best possible technical support to its customers and prospective customers. If you would like personal assistance, please feel free to call us or send e-mail to Customer Service or Technical Support. We are available by phone during regular business hours and in the vast majority of cases, we will return our e-mail with an answer the same day it is received.

	<i>Customer Service</i>	<i>Technical Support</i>
Telephone:	+1 416 920 5884	+1 416 920 2539
E-mail:	<a href="mailto:cs@investintech.com">cs@investintech.com</a>	<a href="mailto:techsupport@investintech.com">techsupport@investintech.com</a>
Hours:	Our Business Hours are Monday to Friday 9am-6pm (Eastern Time GMT-5:00).	

## **Fax and Mailing Address**

Mailing Address: Investintech.com Inc.  
410-96 Spadina Avenue  
Toronto, ON  
M5V 2J6 Canada

Fax: +1 416 920 5848

# **Investintech.com Inc. Software Development Kit User's Guide**

**Investintech PDF2Excel Conversion DLL**

## ***What is the Investintech PDF2Excel Conversion DLL?***

The Investintech PDF2Excel Conversion DLL is a collection of methods compiled, linked and stored in a dynamic-link library (DLL) file. These methods provide a set of methods for converting files from PDF to Microsoft Excel (XLS), and Comma Separated Values (CSV) formats.

## ***Installation Instructions for PDF2ExcelDLL***

Below you will find step-by-step instructions to install PDF2ExcelDLL Conversion Library on your system. You may want to print this instruction sheet for reference before beginning to install PDF2ExcelDLL.

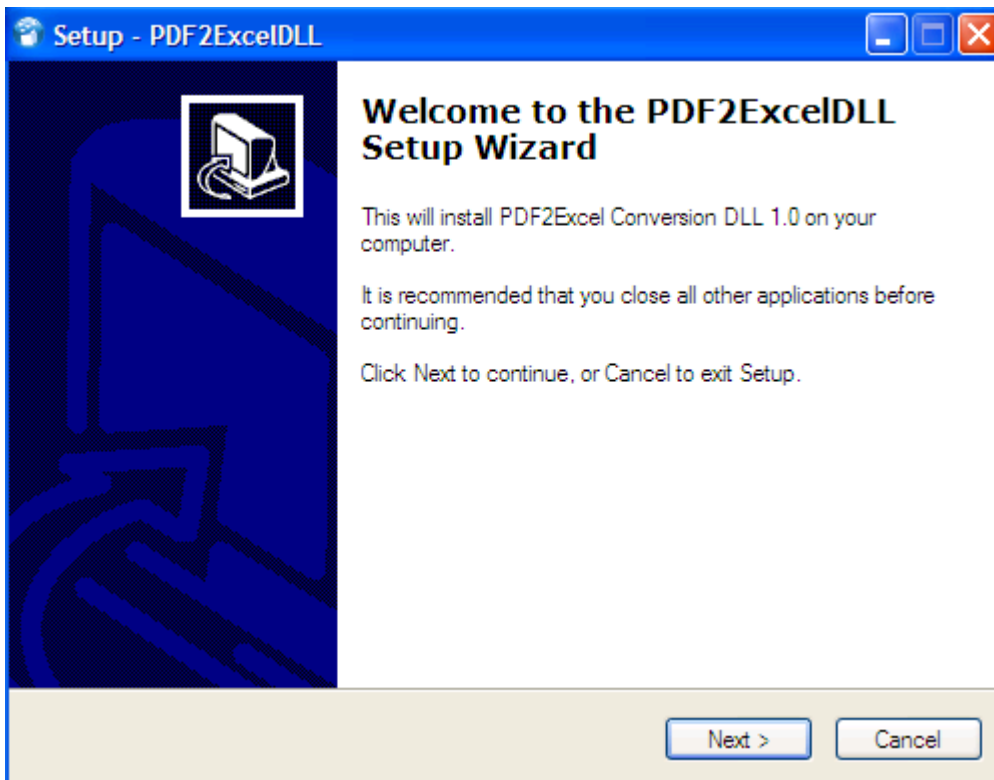
### **System Requirements**

The minimum computer system resources required to install and use PDF2ExcelDLL are:

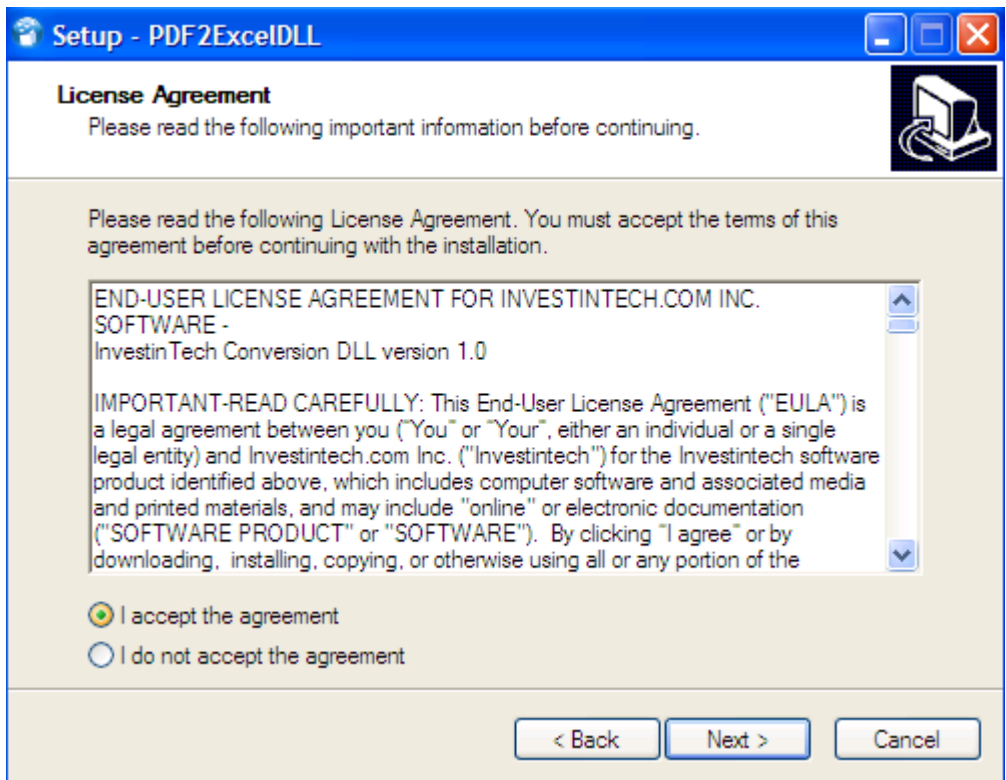
- Microsoft Windows XP
- 64 MB of RAM
- 10 MB of available hard disk space
  - A software development environment, such as Microsoft Visual Studio or Borland Delphi

### **Starting the Installation**

Start the installation by launching the `InstallPDF2ExcelDLL.exe` file, e.g. by double clicking it in the Windows Explorer. After that you should see a window that looks like this:

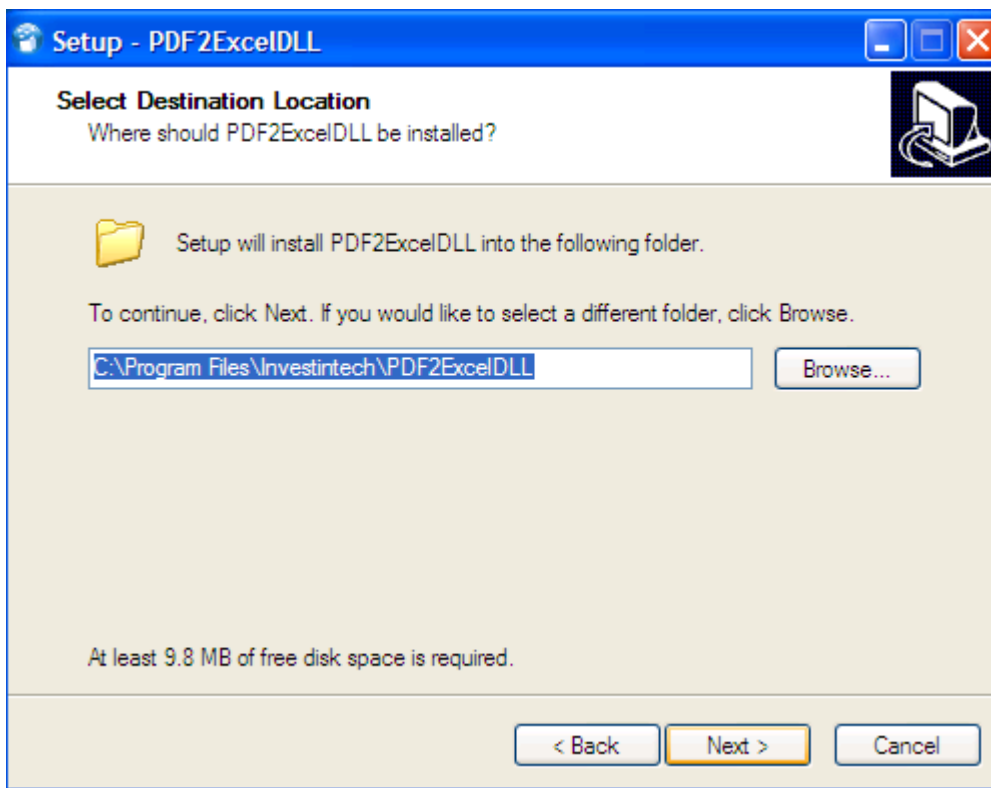


Click the 'Next' button to continue installation. You should see the next window:

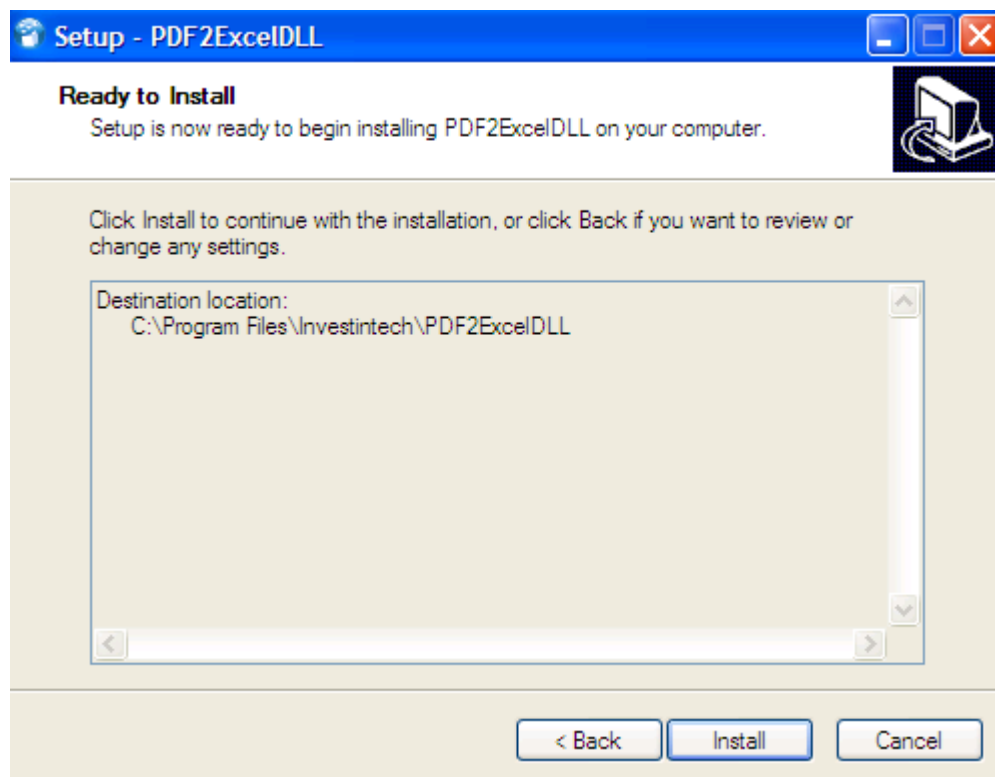


Please read the Investintech Software License Agreement carefully and thoroughly. If you wish to continue with installation, choose **'I agree with the above terms and conditions'** and click **'Next'**. Otherwise, click **'Exit'** and installation will terminate. If you clicked **'Next'** you should see the next window:

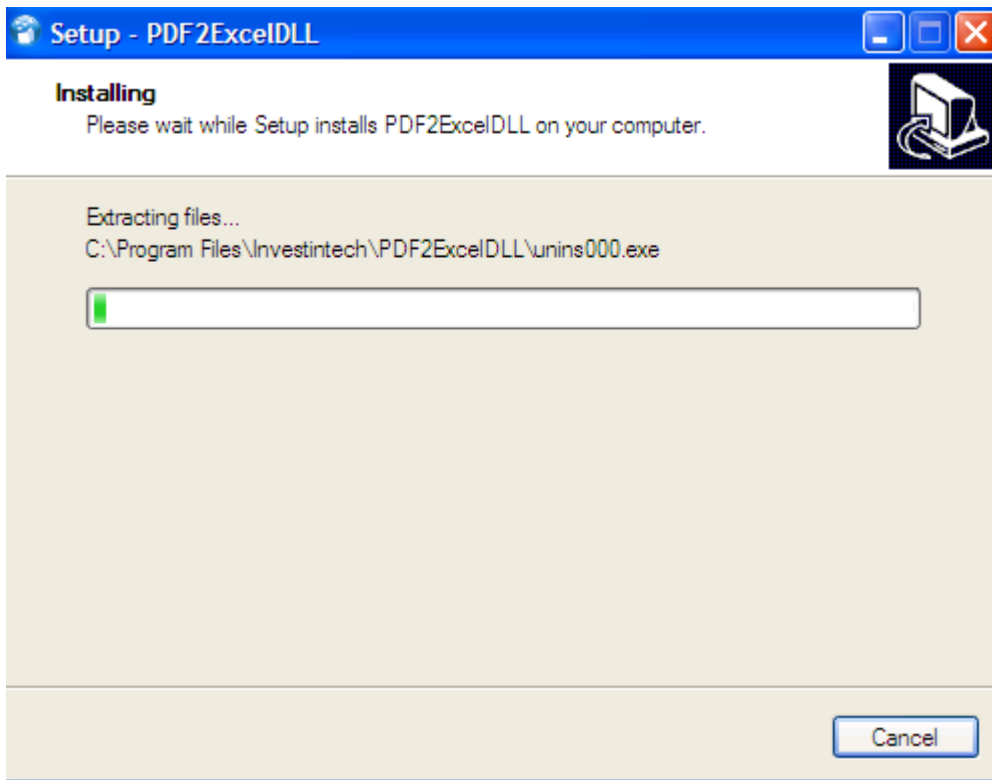




On this screen you can choose the installation directory for the PDF2ExcelDLL software. The default installation folder will be contained in the `Program Files` folder on your system disk. It is possible to change the location of installation folder either by typing its path directly or by clicking the 'Browse...' button. If the destination folder (directory) you have chosen does not exist, the installation program will create it. Click the '**Next**' button to continue the installation.

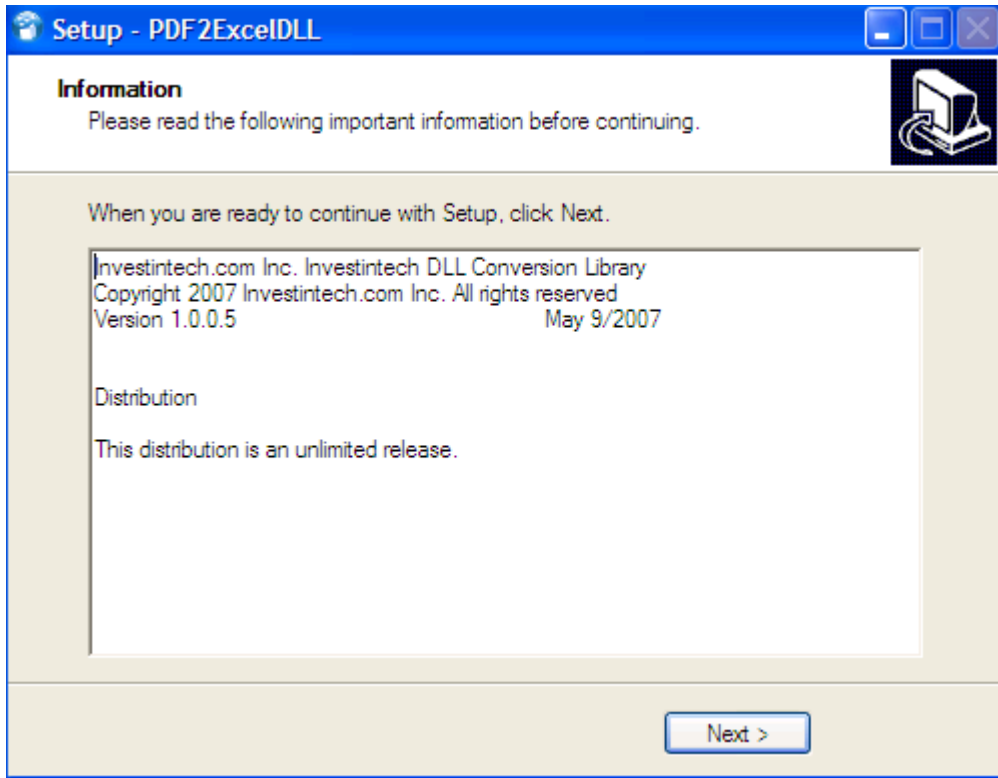


Clicking the **Install** button will start the installation.

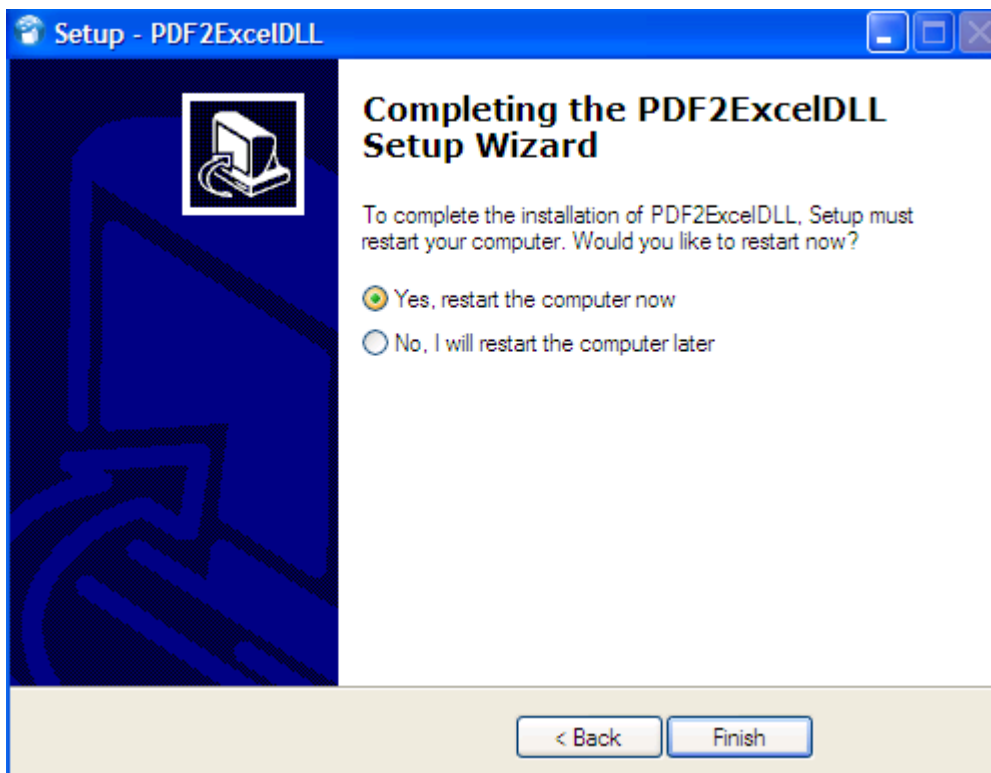


The installation will copy the files that comprise PDF2ExcelDLL to the directory you specified earlier. After a very short period of time, usually about 30 seconds, you should see the next window:

PDF2ExcelDLL – Investintech Conversion DLL



This windows describes the distribution of PDF2ExcelDLL being installed. Click 'Next' to continue.



When this window appears, the installation process is complete. Click **Finish** to end the installation program.

## ***Using the Investintech PDF2Excel Conversion DLL in Microsoft Visual C++ .NET 2003***

In this section you will learn how to use the Investintech PDF2Excel Conversion DLL. All pictures and examples from now on will be based on Microsoft Visual C++ .NET 2003.

In order to use methods contained in DLL you have to link an executable file to DLL. Executable file can be either EXE or DLL file. There are two possible ways of linking: implicit and explicit linking. In this documentation implicit linking will be shown.

### **Implicit Linking**

Implicit linking is load-time linking. The following is needed for executables to implicitly link to Investintech Conversion DLL:

- PDF2ExcelDLL.h, the header file which contains declarations of exported methods
- PDF2ExcelDLL.lib, an import library used by the linker

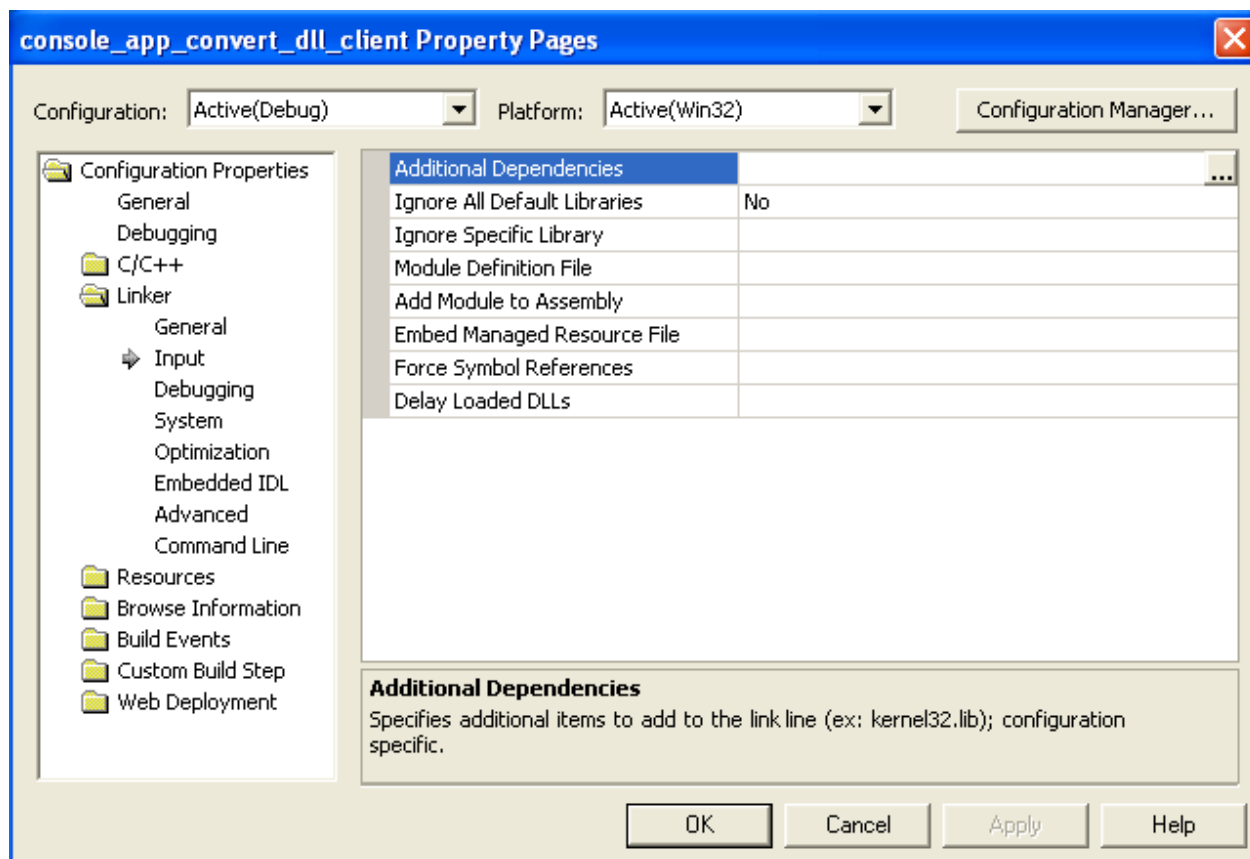
- PDF2ExcelDLL.dll, the Dynamic Link Library file

All these files are provided by Investintech.com Inc.

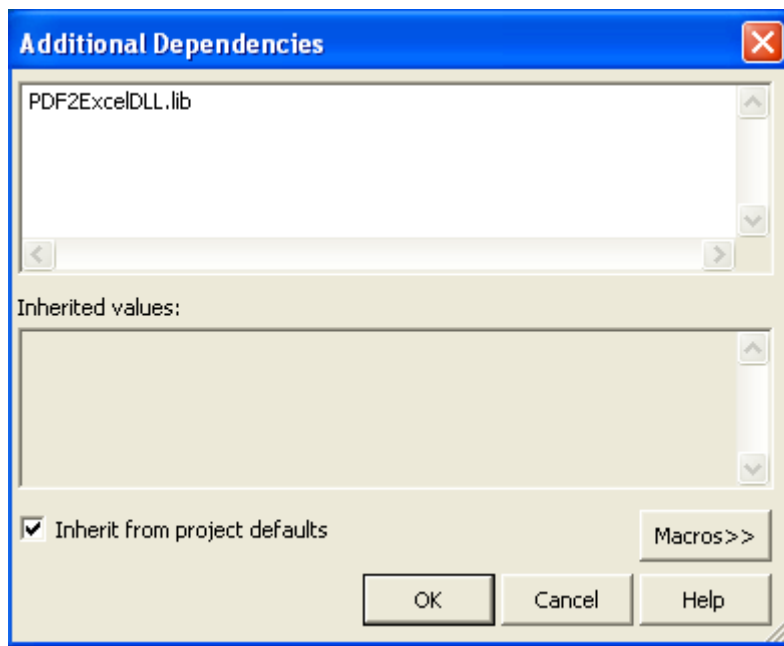
### Linking .lib file with project

Below you will find step-by-step instructions on how to link PDF2ExcelDLL.lib file with your project.

Start Visual Studio and select 'File > Open > Project'. Select the project in which you want to use PDF2ExcelDLL.dll and click 'OK'. Select 'Project > Properties'. You should see the window like next one:



Click the 'Linker' folder and open 'Input' property page. Modify the 'Additional Dependencies' property by clicking the '...' button.



The window titled “Additional Dependencies” should open: type PDF2ExcelDLL.lib and click the ‘OK’ button. Click ‘OK’ to close “Property Pages” window. You have successfully linked PDF2ExcelDLL.lib with your project.

### ***Using methods from DLL file***

After linking with PDF2ExcelDLL.lib file you can use methods contained in InvestintechConversionDLL.dll file. Simply include the PDF2ExcelDLL.h file in all source files that need to use those methods.

## ***Using the Investintech PDF2Excel Conversion DLL in Microsoft Visual C++ .NET 2003***

In this section you will learn how to use the Investintech PDF2Excel Conversion DLL. All pictures and examples from now on will be based on Microsoft Visual C++ .NET 2003.

### ***Investintech PDF2Excel Conversion DLL Methods***

In this section you will find details about each method such as: prototype; description; calling conventions; parameters; return value; and sample use of the methods.

## Interface

The interface of the Investintech PDF2Excel Conversion DLL consists of its exposed methods. The interface contains an exposed method for each combination of: calling convention; parameter type; and error handling.

The calling convention is either the “cdecl” calling convention or the “stdcall” calling convention.

The “cdecl” calling convention is normally used for calling C/C++ functions. It is characterized by the following points:

- arguments are passed in order from right to left
- the calling function is responsible for popping the arguments from the stack at the conclusion of function execution
- an underscore character ( `_` ) is prefixed to function names
- no case translation is performed on the function name

The “stdcall” calling convention is typically used to call Windows operating system API functions. It is characterized by the following points:

- arguments are passed in order from right to left
- arguments are passed by value, unless a pointer or reference type is passed.
- The called function is responsible for popping its own arguments from the stack.
- an underscore ( `_` ) is prefixed to the name. The name is followed by the at sign ( `@` ) followed by the number of bytes (in decimal) in the argument list. For instance, the function declared as `void foo( int bar )` is decorated as follows: `_foo@4`
- no case-translation is performed on the function name

## Parameter Type

The parameter type is either ANSI ( `C char *` ) or UNICODE ( `BSTR` ).

## File Names

Files may be specified by supplying relative or absolute file names. A relative file name is relative to the application default directory containing the application executable file (e.g. “`..\in_parent_dir.pdf`”, “`in_application_dir.pdf`”, “`subdir\in_sub_directory.pdf`”). An absolute file name includes the directory path to the file beginning at the top of the directory tree (e.g. “`c:\pdfs\absolute_file.pdf`”).

It is also possible to use UNC notation (e.g. “`\\myserver\public\file.pdf`”) provided your user account has sufficient access permissions.



## Error Handling

The error handling strategy is either to throw an exception (part of the C++ throw-catch error handling construct) or return in integer/boolean type result code (the C style error handling strategy).

The Investintech Conversion DLL identifies the characteristics of a function by an encoded suffix. The suffix is defined by the rules:

- C or S - cdecl or stdcall calling convention
- B or L - BSTR or LPST parameter type
- R or T - return error code or throw exception

## Common Sample Source Code

All sample code snippets share the following initialization code snippet:

```
//.h file must be included if we want to use conversion methods
#include <afx.h>
#include <afxwin.h>           // MFC core and standard components
#include "PDF2ExcelDLL.h"

CString str_inputFile = _T("c:\pdfs\in.pdf"); //input file name
CString str_templateFile = _T("c:\pdfs\template.ta2e");
CString str_outputFile = _T("out.xls"); //output file name

//conversion from CString to BSTR
BSTR bstr_inputFile = str_inputFile.AllocSysString();
BSTR bstr_templateFile = str_templateFile.AllocString();
BSTR bstr_outputFile = str_outputFile.AllocSysString();
```

## ***Conversion from PDF document to Excel Workbook***

The following methods convert a PDF document file to an Excel workbook file.

## PDF\_to\_Excel\_CBR

### **Prototype**

```
bool PDF_to_Excel_CBR(BSTR inFile, BSTR outFile)
```

### **Description**

The PDF\_to\_Excel\_CBR( ) method converts the user specified file from PDF to Microsoft Excel 2.1 format. Any version of Microsoft Excel can open the created output file. The original PDF file is not modified.

### **Calling Convention**

The function uses the “cdecl” calling convention.

### **Parameters**

<b>inFile</b>	Unicode string containing the absolute or relative filename of the PDF file that will be converted to XLS format.
<b>outFile</b>	Unicode string containing the absolute or relative filename of the XLS file that will contain the result of the conversion.

### **Returns**

true if PDF file is successfully converted to XLS, false otherwise.

### **Example**

```
bool ret = PDF_to_Excel_CBR(bstr_inputFile, bstr_outputFile);
```

## PDF\_to\_Excel\_CBT

### **Prototype**

```
bool PDF_to_Excel_CBT(BSTR inFile, BSTR outFile)
```

### **Description**

The PDF\_to\_Excel\_CBT( ) method converts the user specified file from PDF to Microsoft Excel 2.1 format. Any version of Microsoft Excel can open the created output file. The original PDF file is not modified.

**Calling Convention**

The function uses the “cdecl” calling convention.

**Parameters**

<b>inFile</b>	Unicode string containing the absolute or relative filename of PDF file that will be converted to XLS format.
<b>outFile</b>	Unicode string containing the absolute or relative filename of XLS file containing the result of conversion.

**Returns**

true if PDF file is successfully converted to XLS, otherwise throws an exception that should be handled by the calling application.

**Example**

```
try
{
    bool ret = PDF_to_Excel_CBT(bstr_inputFile, bstr_outputFile);
}
catch(...)
{
    // TODO: Handle exception
}
```

**PDF\_to\_Excel\_CLR****Prototype**

```
bool PDF_to_Excel_CLR(LPSTR inFile, LPSTR outFile)
```

**Description**

The PDF\_to\_Excel\_CLR ( ) method converts the user specified file from PDF to Microsoft Excel 2.1 format. Any version of Microsoft Excel can open the created output file. The original PDF file is not modified.

**Calling Convention**

The function uses the “cdecl” calling convention.

### **Parameters**

- inFile** ANSI string containing the absolute or relative filename of the PDF file that will be converted to XLS format.
- outFile** ANSI string containing the absolute or relative filename of the XLS file containing the result of conversion.

### **Returns**

true if PDF file is successfully converted to XLS, false otherwise.

### **Example**

```
bool ret = PDF_to_Excel_CLR(str_inputFile.GetBuffer(), str_outputFile.GetBuffer());
```

## **PDF\_to\_Excel\_CLT**

### **Prototype**

```
bool PDF_to_Excel_CLT(LPSTR inFile, LPSTR outFile)
```

### **Description**

The PDF\_to\_Excel\_CLT ( ) method converts the user specified file from PDF to Microsoft Excel 2.1 format. Any version of Microsoft Excel can open the created output file. The original PDF file is not modified.

### **Calling Convention**

The function uses the “cdecl” calling convention.

### **Parameters**

- inFile** ANSI string containing the absolute or relative filename of the PDF file that will be converted to XLS format.
- outFile** ANSI string containing the absolute or relative filename of the XLS file containing the result of conversion.

### **Returns**

true if PDF file is successfully converted to XLS, otherwise throws an exception that the calling application should handle.

**Example**

```

try
{
    bool ret = PDF_to_Excel_CLT(str_inputFile.GetBuffer(), str_outputFile.GetBuffer());
}
catch(...)
{
    // TODO: Handle exception
}

```

**PDF\_to\_Excel\_SBR****Prototype**

```

UINT PDF_to_Excel_SBR(BSTR inFile, BSTR outFile)

```

**Description**

The `PDF_to_Excel_SBR ( )` method converts the user specified file from PDF to Microsoft Excel 2.1 format. Any version of Microsoft Excel can open the created output file. The original PDF file is not modified.

**Calling Convention**

The function uses the “stdcall” calling convention.

**Parameters**

<b>inFile</b>	Unicode string containing the absolute or relative filename of the PDF file that will be converted to XLS format.
<b>outFile</b>	Unicode string containing the absolute or relative filename of the XLS file containing the result of conversion.

**Returns**

Non-zero integer value if PDF file is successfully converted to XLS, zero integer value otherwise.

**Example**

```

UINT ret = PDF_to_Excel_SBR(bstr_inputFile, bstr_outputFile);

```

## PDF\_to\_Excel\_SBT

### **Prototype**

```
UINT PDF_to_Excel_SBT(BSTR inFile, BSTR outFile)
```

### **Description**

The PDF\_to\_Excel\_SBT( ) method converts the user specified file from PDF to Microsoft Excel 2.1 format. Any version of Microsoft Excel can open the created output file. The original PDF file is not modified.

### **Calling Convention**

The function uses the “stdcall” calling convention.

### **Parameters**

<b>inFile</b>	Unicode string containing the absolute or relative filename of the PDF file that will be converted to XLS format.
<b>outFile</b>	Unicode string containing the absolute or relative filename of the XLS file containing the result of conversion.

### **Returns**

Non-zero integer value if PDF file is successfully converted to XLS, otherwise throws an exception that the calling application should handle.

### **Example**

```
try
{
    UINT ret = PDF_to_Excel_SBT(bstr_inputFile, bstr_outputFile);
}
catch(...)
{
    // TODO: Handle exception
}
```

## PDF\_to\_Excel\_SLR

### **Prototype**

```
UINT PDF_to_Excel_SLR(LPSTR inFile, LPSTR outFile)
```

**Description**

The PDF\_to\_Excel\_SLR ( ) method converts the user specified file from PDF to Microsoft Excel 2.1 format. Any version of Microsoft Excel can open the created output file. The original PDF file is not modified.

**Calling Convention**

The function uses the “stdcall” calling convention.

**Parameters**

<b>inFile</b>	ANSI string containing the absolute or relative filename of the PDF file that will be converted to XLS format.
<b>outFile</b>	ANSI string containing the absolute or relative filename of the XLS file containing the result of conversion.

**Returns**

Non-zero integer value if PDF file is successfully converted to XLS, zero integer value otherwise.

**Example**

```
UINT retCode = PDF_to_Excel_SLR(str_inputFile, str_outputFile);
```

**PDF\_to\_Excel\_SLT****Prototype**

```
UINT PDF_to_Excel_SLT(LPSTR inFile, LPSTR outFile)
```

**Description**

The PDF\_to\_Excel\_SLT ( ) method converts the user specified file from PDF to Microsoft Excel 2.1 format. Any version of Microsoft Excel can open the created output file. The original PDF file is not modified.

**Calling Convention**

The function uses the “stdcall” calling convention.

### **Parameters**

- inFile** ANSI string containing absolute or relative filename of the PDF file that will be converted to XLS format.
- outFile** ANSI string containing the absolute or relative filename of the XLS file containing the result of conversion.

### **Returns**

Non-zero integer value if PDF file is successfully converted to XLS, otherwise throws an exception that should be handled by the calling application.

### **Example**

```
try
{
    UINT retCode = PDF_to_Excel_SLT(str_inputFile, str_outputFile);
}
catch(...)
{
    //TODO: handle error
}
```

## ***Conversion from PDF document to Excel Workbook using a template file***

The following methods convert a PDF document file to an Excel workbook file using a template file to describe column positions in the PDF file.

### **PDF\_to\_Excel\_Template\_CBR**

#### **Prototype**

```
bool PDF_to_Excel_Template_CBR(BSTR inFile, BSTR outFile, BSTR
templateFile)
```

#### **Description**

The PDF\_to\_Excel\_Template\_CBR() method converts the user specified file from PDF to Microsoft Excel 2.1 format. It uses the column positions described in the template file to convert data on the PDF file. Any version of Microsoft Excel can open the created output file. The original PDF file is not modified.

#### **Calling Convention**

The function uses the “cdecl” calling convention.



**Parameters**

- inFile** Unicode string containing the absolute or relative filename of the PDF file that will be converted to XLS format.
- outFile** Unicode string containing the absolute or relative filename of the XLS file containing the result of conversion.
- templateFile** Unicode string containing the absolute or relative filename of the template file used to parse the PDF file.

**Returns**

true if PDF file is successfully converted to XLS, false otherwise.

**Example**

```
bool ret = PDF_to_Excel_Template_CBR(str_inputFile, str_outputFile, str_templateFile);
```

**PDF\_to\_Excel\_Template\_CBT****Prototype**

```
bool PDF_to_Excel_Template_CBT(BSTR inFile, BSTR outFile, BSTR
templateFile)
```

**Description**

The `PDF_to_Excel_Template_CBT()` method converts the user specified file from PDF to Microsoft Excel 2.1 format. It uses the column positions described in the template file to convert data on the PDF file. Any version of Microsoft Excel can open the created output file. The original PDF file is not modified.

**Calling Convention**

The function uses the “cdecl” calling convention.

**Parameters**

- inFile** Unicode string containing the absolute or relative filename of the PDF file that will be converted to XLS format.
- outFile** Unicode string containing the absolute or relative filename of the XLS file containing the result of conversion.
- templateFile** Unicode string containing the absolute or relative filename of the template file used to parse the PDF file.

### **Returns**

true if PDF file is successfully converted to XLS, otherwise throws an exception that should be handled by the calling application.

### **Example**

```
try
{
    bool ret = PDF_to_Excel_Template_CBT(str_inputFile, str_outputFile, str_templateFile);
}
catch(...)
{
    //TODO: handle error
}
```

## **PDF\_to\_Excel\_Template\_CLR**

### **Prototype**

```
bool PDF_to_Excel_Template_CLR(LPSTR inFile, LPSTR outFile, LPSTR
templateFile)
```

### **Description**

The PDF\_to\_Excel\_Template\_CLR() method converts the user specified file from PDF to Microsoft Excel 2.1 format. It uses the column positions described in the template file to convert data on the PDF file. Any version of Microsoft Excel can open the created output file. The original PDF file is not modified.

### **Calling Convention**

The function uses the “cdecl” calling convention.

### **Parameters**

- |                     |  |
|---------------------|--|
| <b>inFile</b>       | ANSI string containing the absolute or relative filename of the PDF file that will be converted to XLS format. |
| <b>outFile</b>      | ANSI string containing the absolute or relative filename of the XLS file containing the result of conversion.  |
| <b>templateFile</b> | ANSI string containing the absolute or relative filename of the template file used to parse the PDF file.      |

**Returns**

true if PDF file is successfully converted to XLS, false otherwise.

**Example**

```
bool ret = PDF_to_Excel_Template_CLR(str_inputFile, str_outputFile, str_templateFile);
```

**PDF\_to\_Excel\_Template\_CLT****Prototype**

```
bool PDF_to_Excel_Template_CLT(LPSTR inFile, LPSTR outFile, LPSTR
templateFile)
```

**Description**

The PDF\_to\_Excel\_Template\_CLT() method converts the user specified file from PDF to Microsoft Excel 2.1 format. It uses the column positions described in the template file to convert data on the PDF file. Any version of Microsoft Excel can open the created output file. The original PDF file is not modified.

**Calling Convention**

The function uses the “cdecl” calling convention.

**Parameters**

- |                     |  |
|---------------------|--|
| <b>inFile</b>       | ANSI string containing the absolute or relative filename of the PDF file that will be converted to XLS format. |
| <b>outFile</b>      | ANSI string containing the absolute or relative filename of the XLS file containing the result of conversion.  |
| <b>templateFile</b> | ANSI string containing the absolute or relative filename of the template file used to parse the PDF file.      |

**Returns**

true if PDF file is successfully converted to XLS, otherwise throws an exception that should be handled by the calling application.

**Example**

```
try
{
    bool ret = PDF_to_Excel_Template_CLT(str_inputFile, str_outputFile, str_templateFile);
```

```
}  
catch(...)  
{  
    //TODO: handle error  
}
```

-----

## PDF\_to\_Excel\_Template\_SBR

### **Prototype**

```
UINT PDF_to_Excel_Template_SBR(BSTR inFile, BSTR outFile, BSTR  
templateFile)
```

### **Description**

The PDF\_to\_Excel\_Template\_SBR() method converts the user specified file from PDF to Microsoft Excel 2.1 format. It uses the column positions described in the template file to convert data on the PDF file. Any version of Microsoft Excel can open the created output file. The original PDF file is not modified.

### **Calling Convention**

The function uses the “stdcall” calling convention.

### **Parameters**

- inFile** Unicode string containing the absolute or relative filename of the PDF file that will be converted to XLS format.
- outFile** Unicode string containing the absolute or relative filename of the XLS file containing the result of conversion.
- templateFile** Unicode string containing the absolute or relative filename of the template file used to parse the PDF file.

### **Returns**

Non-zero integer value if PDF file is successfully converted to XLS, zero integer value otherwise.

### **Example**

```
UINT ret = PDF_to_Excel_Template_SBR(str_inputFile, str_outputFile, str_templateFile);
```

## PDF\_to\_Excel\_Template\_SBT

### **Prototype**

```
UINT PDF_to_Excel_Template_SBT(BSTR inFile, BSTR outFile, BSTR
templateFile)
```

### **Description**

The PDF\_to\_Excel\_Template\_SBT() method converts the user specified file from PDF to Microsoft Excel 2.1 format. It uses the column positions described in the template file to convert data on the PDF file. Any version of Microsoft Excel can open the created output file. The original PDF file is not modified.

### **Calling Convention**

The function uses the “stdcall” calling convention.

### **Parameters**

<b>inFile</b>	Unicode string containing the absolute or relative filename of the PDF file that will be converted to XLS format.
<b>outFile</b>	Unicode string containing the absolute or relative filename of the XLS file containing the result of conversion.
<b>templateFile</b>	Unicode string containing the absolute or relative filename of the template file used to parse the PDF file.

### **Returns**

Non-zero integer value if PDF file is successfully converted to XLS, otherwise throws an exception that should be handled by the calling application.

### **Example**

```
try
{
    UINT ret = PDF_to_Excel_Template_SBT(bstr_inputFile, bstr_outputFile, bstr_templateFile);
}
catch(...)
{
    //TODO: handle error
}
```

## PDF\_to\_Excel\_Template\_SLR

### **Prototype**

```
UINT PDF_to_Excel_Template_SLR(LPSTR inFile, LPSTR outFile, LPSTR  
templateFile)
```

### **Description**

The PDF\_to\_Excel\_Template\_SLR() method converts the user specified file from PDF to Microsoft Excel 2.1 format. It uses the column positions described in the template file to convert data on the PDF file. Any version of Microsoft Excel can open the created output file. The original PDF file is not modified.

### **Calling Convention**

The function uses the “stdcall” calling convention.

### **Parameters**

- inFile** ANSI string containing the absolute or relative filename of the PDF file that will be converted to XLS format.
- outFile** ANSI string containing the absolute or relative filename of the XLS file containing the result of conversion.
- templateFile** Unicode string containing the absolute or relative filename of the template file used to parse the PDF file.

### **Returns**

Non-zero integer value if PDF file is successfully converted to XLS, zero integer value otherwise.

### **Example**

```
UINT ret = PDF_to_Excel_Template_SLR(str_inputFile, str_outputFile, str_templateFile);
```

## PDF\_to\_Excel\_Template\_SLT

### **Prototype**

```
UINT PDF_to_Excel_Template_SLT(LPSTR inFile, LPSTR outFile, LPSTR  
templateFile)
```

### **Description**

The PDF\_to\_Excel\_Template\_SLT() method converts the user specified file from PDF to Microsoft Excel 2.1 format. It uses the column positions described in the template file to convert data on the PDF file. Any version of Microsoft Excel can open the created output file. The original PDF file is not modified.

### **Calling Convention**

The function uses the “stdcall” calling convention.

### **Parameters**

- inFile** ANSI string containing the absolute or relative filename of the PDF file that will be converted to XLS format.
- outFile** ANSI string containing the absolute or relative filename of the XLS file containing the result of conversion.
- templateFile** ANSI string containing the absolute or relative filename of the template file used to parse the PDF file.

### **Returns**

Non-zero integer value if PDF file is successfully converted to XLS, otherwise throws an exception that should be handled by the calling application.

### **Example**

```
try  
{  
    UINT ret = PDF_to_Excel_Template_SLT(str_inputFile, str_outputFile, str_templateFile);  
}  
catch(...)  
{  
    //TODO: handle error  
}
```

## **Convert from PDF document to CSV file**

The following methods convert a PDF document file to a Comma Separated Value text file.

### **PDF\_to\_CSV\_CBR**

#### **Prototype**

```
bool PDF_to_CSV_CBR(BSTR inFile, BSTR outFile)
```

#### **Description**

The PDF\_to\_CSV\_CBR() method converts the user specified file from PDF to Comma Separated Value format. The created output file may be opened with Microsoft Excel or a text editor. The original PDF file is not modified.

#### **Calling Convention**

The function uses the “cdecl” calling convention.

#### **Parameters**

<b>inFile</b>	Unicode string containing the absolute or relative filename of the PDF file that will be converted to CSV format.
<b>outFile</b>	Unicode string containing the absolute or relative filename of the CSV file that will contain the result of the conversion.

#### **Returns**

true if PDF file is successfully converted to CSV, false otherwise.

#### **Example**

```
bool ret = PDF_to_CSV_CBR(bstr_inputFile, bstr_outputFile);
```

### **PDF\_to\_CSV\_CBT**

#### **Prototype**

```
bool PDF_to_CSV_CBT(BSTR inFile, BSTR outFile)
```



**Description**

The `PDF_to_CSV_CBT()` method converts the user specified file from PDF to Comma Separated Value format. The created output file may be opened with Microsoft Excel or a text editor. The original PDF file is not modified.

**Calling Convention**

The function uses the “cdecl” calling convention.

**Parameters**

<b>inFile</b>	Unicode string containing the absolute or relative filename of PDF file that will be converted to CSV format.
<b>outFile</b>	Unicode string containing the absolute or relative filename of CSV file containing the result of conversion.

**Returns**

`true` if PDF file is successfully converted to CSV, otherwise throws an exception that should be handled by the calling application.

**Example**

```
try
{
    bool ret = PDF_to_CSV_CBT(bstr_inputFile, bstr_outputFile);
}
catch(...)
{
    // TODO: Handle exception
}
```

**PDF\_to\_CSV\_CLR****Prototype**

```
bool PDF_to_CSV_CLR(LPSTR inFile, LPSTR outFile)
```

### **Description**

The PDF\_to\_CSV\_CLR() method converts the user specified file from PDF to Comma Separated Value format. The created output file may be opened with Microsoft Excel or a text editor. The original PDF file is not modified.

### **Calling Convention**

The function uses the “cdecl” calling convention.

### **Parameters**

- inFile**           ANSI string containing the absolute or relative filename of the PDF file that will be converted to CSV format.
- outFile**          ANSI string containing the absolute or relative filename of the CSV file that will contain the result of the conversion.

### **Returns**

true if PDF file is successfully converted to CSV, false otherwise.

### **Example**

```
bool ret = PDF_to_CSV_CLR(str_inputFile, str_outputFile);
```

## **PDF\_to\_CSV\_CLT**

### **Prototype**

```
bool PDF_to_CSV_CLT(LPSTR inFile, LPSTR outFile)
```

### **Description**

The PDF\_to\_CSV\_CLT() method converts the user specified file from PDF to Comma Separated Value format. The created output file may be opened with Microsoft Excel or a text editor. The original PDF file is not modified.

### **Calling Convention**

The function uses the “cdecl” calling convention.

**Parameters**

<b>inFile</b>	ANSI string containing the absolute or relative filename of PDF file that will be converted to CSV format.
<b>outFile</b>	ANSI string containing the absolute or relative filename of CSV file containing the result of conversion.

**Returns**

true if PDF file is successfully converted to CSV, otherwise throws an exception that should be handled by the calling application.

**Example**

```
try
{
    bool ret = PDF_to_CSV_CLT(str_inputFile, str_outputFile);
}
catch(...)
{
    // TODO: Handle exception
}
```

**PDF\_to\_CSV\_SBR****Prototype**

```
bool PDF_to_CSV_SBR(BSTR inFile, BSTR outFile)
```

**Description**

The PDF\_to\_CSV\_SBR() method converts the user specified file from PDF to Comma Separated Value format. The created output file may be opened with Microsoft Excel or a text editor. The original PDF file is not modified.

**Calling Convention**

The function uses the “stdcall” calling convention.

**Parameters**

<b>inFile</b>	Unicode string containing the absolute or relative filename of the PDF file that will be converted to CSV format.
---------------	---

**outFile** Unicode string containing the absolute or relative filename of the CSV file that will contain the result of the conversion.

### **Returns**

Non-zero integer value if PDF file is successfully converted to CSV, zero integer value otherwise.

### **Example**

```
UINT ret = PDF_to_CSB_SBR(bstr_inputFile, bstr_outputFile);
```

## **PDF\_to\_CSV\_SBT**

### **Prototype**

```
bool PDF_to_CSV_SBT(BSTR inFile, BSTR outFile)
```

### **Description**

The PDF\_to\_CSV\_SBT() method converts the user specified file from PDF to Comma Separated Value format. The created output file may be opened with Microsoft Excel or a text editor. The original PDF file is not modified.

### **Calling Convention**

The function uses the “stdcall” calling convention.

### **Parameters**

**inFile** Unicode string containing the absolute or relative filename of PDF file that will be converted to CSV format.

**outFile** Unicode string containing the absolute or relative filename of CSV files containing the result of conversion.

### **Returns**

Non-zero integer value if PDF file is successfully converted to CSV, otherwise throws an exception that should be handled by the calling application.

### **Example**

```
try  
{
```

```

    bool ret = PDF_to_CSV_SBT(bstr_inputFile, bstr_outputFile);
}
catch(...)
{
    // TODO: Handle exception
}

```

## PDF\_to\_CSV\_SLR

### *Prototype*

```
bool PDF_to_CSV_SLR(LPSTR inFile, LPSTR outFile)
```

### *Description*

The PDF\_to\_CSV\_SLR() method converts the user specified file from PDF to Comma Separated Value format. The created output file may be opened with Microsoft Excel or a text editor. The original PDF file is not modified.

### *Calling Convention*

The function uses the “stdcall” calling convention.

### *Parameters*

<b>inFile</b>	ANSI string containing the absolute or relative filename of the PDF file that will be converted to CSV format.
<b>outFile</b>	ANSI string containing the absolute or relative filename of the CSV file that will contain the result of the conversion.

### *Returns*

Non-zero integer value if PDF file is successfully converted to CSV, zero integer value otherwise.

### *Example*

```
UINT ret = PDF_to_CSV_SLR(str_inputFile, str_outputFile);
```

## PDF\_to\_CSV\_SLT

### **Prototype**

```
bool PDF_to_CSV_SLT(LPSTR inFile, LPSTR outFile)
```

### **Description**

The PDF\_to\_CSV\_SLT() method converts the user specified file from PDF to Comma Separated Value format. The created output file may be opened with Microsoft Excel or a text editor. The original PDF file is not modified.

### **Calling Convention**

The function uses the “stdcall” calling convention.

### **Parameters**

<b>inFile</b>	ANSI string containing the absolute or relative filename of PDF file that will be converted to CSV format.
<b>outFile</b>	ANSI string containing the absolute or relative filename of CSV files containing the result of conversion.

### **Returns**

Non-zero integer value if PDF file is successfully converted to CSV, otherwise throws an exception that should be handled by the calling application.

### **Example**

```
try
{
    bool ret = PDF_to_CSV_SLT(str_inputFile, str_outputFile);
}
catch(...)
{
    // TODO: Handle exception
}
```

## **Convert from PDF document to CSV file using template file**

The following methods convert a PDF document file to an Excel workbook file using a template file to describe column positions in the PDF file.

## PDF\_to\_CSV\_Template\_CBR

### **Prototype**

```
bool PDF_to_CSV_Template_CBR(BSTR inFile, BSTR outFile, BSTR  
templateFile)
```

### **Description**

The PDF\_to\_CSV\_Template\_CBR() method converts the user specified file from PDF to Comma Separated Value format. It uses the column positions described in the template file to convert data on the PDF file. The output file may be opened with a text editor or any version of Microsoft Excel. The original PDF file is not modified.

### **Calling Convention**

The function uses the “cdecl” calling convention.

### **Parameters**

<b>inFile</b>	Unicode string containing the absolute or relative filename of the PDF file that will be converted to CSV format.
<b>outFile</b>	Unicode string containing the absolute or relative filename of the CSV file containing the result of conversion.
<b>templateFile</b>	Unicode string containing the absolute or relative filename of the template file used to parse the PDF file.

### **Returns**

true if PDF file is successfully converted to XLS, false otherwise.

### **Example**

```
bool ret = PDF_to_CSV_Template_CBR(bstr_inputFile, bstr_outputFile, bstr_templateFile);
```

## PDF\_to\_CSV\_Template\_CBT

### **Prototype**

```
bool PDF_to_CSV_Template_CBT(BSTR inFile, BSTR outFile, BSTR  
templateFile)
```

### **Description**

The `PDF_to_CSV_Template_CBT()` method converts the user specified file from PDF to Comma Separated Value format. It uses the column positions described in the template file to convert data on the PDF file. The output file may be opened with a text editor or any version of Microsoft Excel. The original PDF file is not modified.

### **Calling Convention**

The function uses the “cdecl” calling convention.

### **Parameters**

- inFile** Unicode string containing the absolute or relative filename of the PDF file that will be converted to XLS format.
- outFile** Unicode string containing the absolute or relative filename of the XLS file containing the result of conversion.
- templateFile** Unicode string containing the absolute or relative filename of the template file used to parse the PDF file.

### **Returns**

true if PDF file is successfully converted to CSV, otherwise throws an exception that should be handled by the calling application.

### **Example**

```
try
{
    bool ret = PDF_to_CSV_Template_CBT(bstr_inputFile, bstr_outputFile, bstr_templateFile);
}
catch(...)
{
    //TODO: handle error
}
```

## **PDF\_to\_CSV\_Template\_CLR**

### **Prototype**

```
bool PDF_to_CSV_Template_CLR(LPSTR inFile, LPSTR outFile, LPSTR
templateFile)
```



**Description**

The `PDF_to_CSV_Template_CLR()` method converts the user specified file from PDF to Comma Separated Value format. It uses the column positions described in the template file to convert data on the PDF file. The output file may be opened with a text editor or any version of Microsoft Excel. The original PDF file is not modified.

**Calling Convention**

The function uses the “cdecl” calling convention.

**Parameters**

<b>inFile</b>	ANSI string containing the absolute or relative filename of the PDF file that will be converted to CSV format.
<b>outFile</b>	ANSI string containing the absolute or relative filename of the CSV file containing the result of conversion.
<b>templateFile</b>	ANSI string containing the absolute or relative filename of the template file used to parse the PDF file.

**Returns**

true if PDF file is successfully converted to CSV, false otherwise.

**Example**

```
bool ret = PDF_to_CSV_Template_CLR(str_inputFile, str_outputFile, str_templateFile);
```

**PDF\_to\_CSV\_Template\_CLT****Prototype**

```
bool PDF_to_CSV_Template_CLT(LPSTR inFile, LPSTR outFile, LPSTR
templateFile)
```

**Description**

The `PDF_to_CSV_Template_CLT()` method converts the user specified file from PDF to Comma Separated Value format. It uses the column positions described in the template file to convert data on the PDF file. The output file may be opened with a text editor or any version of Microsoft Excel. The original PDF file is not modified.

**Calling Convention**

The function uses the “cdecl” calling convention.

### **Parameters**

- inFile** ANSI string containing the absolute or relative filename of the PDF file that will be converted to XLS format.
- outFile** ANSI string containing the absolute or relative filename of the XLS file containing the result of conversion.
- templateFile** ANSI string containing the absolute or relative filename of the template file used to parse the PDF file.

### **Returns**

true if PDF file is successfully converted to CSV, otherwise throws an exception that should be handled by the calling application.

### **Example**

```
try
{
    bool ret = PDF_to_CSV_Template_CLT(str_inputFile, str_outputFile, str_templateFile);
}
catch(...)
{
    //TODO: handle error
}
```

## **PDF\_to\_CSV\_Template\_SBR**

### **Prototype**

```
UINT PDF_to_CSV_Template_SBR(BSTR inFile, BSTR outFile, BSTR
templateFile)
```

### **Description**

The PDF\_to\_CSV\_Template\_SBR() method converts the user specified file from PDF to Comma Separated Value format. It uses the column positions described in the template file to convert data on the PDF file. The output file may be opened with a text editor or any version of Microsoft Excel. The original PDF file is not modified.

### **Calling Convention**

The function uses the “stdcall” calling convention.

**Parameters**

- inFile** Unicode string containing the absolute or relative filename of the PDF file that will be converted to CSV format.
- outFile** Unicode string containing the absolute or relative filename of the CSV file containing the result of conversion.
- templateFile** Unicode string containing the absolute or relative filename of the template file used to parse the PDF file.

**Returns**

Non-zero integer value if PDF file is successfully converted to XLS, zero integer value otherwise.

**Example**

```
UINT ret = PDF_to_CSV_Template_SBR(bstr_inputFile, bstr_outputFile, bstr_templateFile);
```

**PDF\_to\_CSV\_Template\_SBT****Prototype**

```
UINT PDF_to_CSV_Template_SBT(BSTR inFile, BSTR outFile, BSTR
templateFile)
```

**Description**

The PDF\_to\_CSV\_Template\_SBT() method converts the user specified file from PDF to Comma Separated Value format. It uses the column positions described in the template file to convert data on the PDF file. The output file may be opened with a text editor or any version of Microsoft Excel. The original PDF file is not modified.

**Calling Convention**

The function uses the “stdcall” calling convention.

**Parameters**

- inFile** Unicode string containing the absolute or relative filename of the PDF file that will be converted to XLS format.
- outFile** Unicode string containing the absolute or relative filename of the XLS file containing the result of conversion.
- templateFile** Unicode string containing the absolute or relative filename of the template file used to parse the PDF file.

### **Returns**

true if PDF file is successfully converted to CSV, otherwise throws an exception that should be handled by the calling application.

### **Example**

```
try
{
    UINT ret = PDF_to_CSV_Template_CBT(str_inputFile, str_outputFile, str_templateFile);
}
catch(...)
{
    //TODO: handle error
}
```

## **PDF\_to\_CSV\_Template\_SLR**

### **Prototype**

```
UINT PDF_to_CSV_Template_SLR(LPSTR inFile, BSTR outFile, BSTR
templateFile)
```

### **Description**

The PDF\_to\_CSV\_Template\_SLR() method converts the user specified file from PDF to Comma Separated Value format. It uses the column positions described in the template file to convert data on the PDF file. The output file may be opened with a text editor or any version of Microsoft Excel. The original PDF file is not modified.

### **Calling Convention**

The function uses the “stdcall” calling convention.

### **Parameters**

- |                     |  |
|---------------------|--|
| <b>inFile</b>       | ANSI string containing the absolute or relative filename of the PDF file that will be converted to CSV format. |
| <b>outFile</b>      | ANSI string containing the absolute or relative filename of the CSV file containing the result of conversion.  |
| <b>templateFile</b> | ANSI string containing the absolute or relative filename of the template file used to parse the PDF file.      |

**Returns**

Non-zero integer value if PDF file is successfully converted to CSV, zero integer value otherwise.

**Example**

```
UINT ret = PDF_to_CSV_Template_SLR(str_inputFile, str_outputFile, bstr_templateFile);
```

**PDF\_to\_CSV\_Template\_SLT****Prototype**

```
UINT PDF_to_CSV_Template_SLT(LPSTR inFile, LPSTR outFile, LPSTR  
templateFile)
```

**Description**

The PDF\_to\_CSV\_Template\_SLT() method converts the user specified file from PDF to Comma Separated Value format. It uses the column positions described in the template file to convert data on the PDF file. The output file may be opened with a text editor or any version of Microsoft Excel. The original PDF file is not modified.

**Calling Convention**

The function uses the “stdcall” calling convention.

**Parameters**

- |                     |  |
|---------------------|--|
| <b>inFile</b>       | ANSI string containing the absolute or relative filename of the PDF file that will be converted to XLS format. |
| <b>outFile</b>      | ANSI string containing the absolute or relative filename of the XLS file containing the result of conversion.  |
| <b>templateFile</b> | ANSI string containing the absolute or relative filename of the template file used to parse the PDF file.      |

**Returns**

Non-zero integer value if PDF file is successfully converted to CSV, otherwise throws an exception that the calling application should handle.

### **Example**

```
try
{
    bool ret = PDF_to_CSV_Template_SLT(str_inputFile, str_outputFile, str_templateFile);
}
catch(...)
{
    //TODO: handle error
}
```

### **VB6 callable code**

VB6 client applications must use these functions in order to avoid runtime errors related to stack and string references.

## **PDF\_to\_Excel\_VB6**

### **Prototype**

```
UINT PDF_to_Excel_VB6(BSTR inFile, BSTR outFile)
```

### **Description**

The PDF\_to\_Excel\_VB6 ( ) method converts the user specified file from PDF to Microsoft Excel 2.1 format. Any version of Microsoft Excel can open the created output file. The original PDF file is not modified.

### **Calling Convention**

The function uses the “stdcall” calling convention.

### **Parameters**

- |                |   |
|----------------|---|
| <b>inFile</b>  | Unicode string containing the absolute or relative filename of the PDF file that will be converted to XLS format. |
| <b>outFile</b> | Unicode string containing the absolute or relative filename of the XLS file containing the result of conversion.  |

**Returns**

Non-zero integer value if PDF file is successfully converted to XLS, zero integer value otherwise.

**Example**

```
UINT ret = PDF_to_Excel_VB6(bstr_inputFile, bstr_outputFile);
```

**PDF\_to\_Excel\_Template\_VB6****Prototype**

```
UINT PDF_to_Excel_Template_VB6(BSTR inFile, BSTR outFile, BSTR
templateFile)
```

**Description**

The PDF\_to\_Excel\_Template\_VB6() method converts the user specified file from PDF to Microsoft Excel 2.1 format. It uses the column positions described in the template file to convert data on the PDF file. Any version of Microsoft Excel can open the created output file. The original PDF file is not modified.

**Calling Convention**

The function uses the “stdcall” calling convention.

**Parameters**

<b>inFile</b>	Unicode string containing the absolute or relative filename of the PDF file that will be converted to XLS format.
<b>outFile</b>	Unicode string containing the absolute or relative filename of the XLS file containing the result of conversion.
<b>templateFile</b>	Unicode string containing the absolute or relative filename of the template file used to parse the PDF file.

**Returns**

Non-zero integer value if PDF file is successfully converted to XLS, zero integer value otherwise.

**Example**

```
try
{
```

```
    UINT ret = PDF_to_Excel_Template_VB6(str_inputFile, str_outputFile, str_templateFile);  
}  
catch(...)  
{  
    //TODO: handle error  
}
```

PDF\_to\_CSV\_VB6

### **Prototype**

```
UINT PDF_to_CSV_VB6(BSTR inFile, BSTR outFile)
```

### **Description**

The PDF\_to\_CSV\_VB6() method converts the user specified file from PDF to Comma Separated Value format. The created output file may be opened with Microsoft Excel or a text editor. The original PDF file is not modified.

### **Calling Convention**

The function uses the “stdcall” calling convention.

### **Parameters**

<b>inFile</b>	Unicode string containing the absolute or relative filename of the PDF file that will be converted to CSV format.
<b>outFile</b>	Unicode string containing the absolute or relative filename of the CSV file that will contain the result of the conversion.

### **Returns**

Non-zero integer value if PDF file is successfully converted to CSV, zero integer value otherwise.

### **Example**

```
    UINT ret = PDF_to_CSV_VB6(bstr_inputFile, bstr_outputFile);
```



## PDF\_to\_CSV\_Template\_VB6

### **Prototype**

```
UINT PDF_to_CSV_Template_VB6(BSTR inFile, BSTR outFile, BSTR
templateFile)
```

### **Description**

The `PDF_to_CSV_Template_VB6()` method converts the user specified file from PDF to Comma Separated Value format. It uses the column positions described in the template file to convert data on the PDF file. The output file may be opened with a text editor or any version of Microsoft Excel. The original PDF file is not modified.

### **Calling Convention**

The function uses the “stdcall” calling convention.

### **Parameters**

<b>inFile</b>	Unicode string containing the absolute or relative filename of the PDF file that will be converted to CSV format.
<b>outFile</b>	Unicode string containing the absolute or relative filename of the CSV file containing the result of conversion.
<b>templateFile</b>	Unicode string containing the absolute or relative filename of the template file used to parse the PDF file.

### **Returns**

Non-zero integer value if PDF file is successfully converted to XLS, zero integer value otherwise.

### **Example**

```
UINT ret = PDF_to_CSV_Template_VB6(bstr_inputFile, bstr_outputFile, bstr_templateFile);
```

## InvestintechConversionDLL\_IDString

```
char* InvestintechConversionDLL_IDString()
```

### **Description**

The `InvestintechConversionDLL_IDString()` method returns vendor’s ID string.

PDF2ExcelDLL – Investintech Conversion DLL

***Parameters***

None.

***Returns***

"Investintech Conversion DLL Version 3.00 by Investintech.com Inc." string.

## ***Investintech PDF-To-Excel Conversion COM Server Methods***

This section provides a description of the conversion methods exposed by the PDF-To-Excel COM Server. The COM Server defines the CoClass CPDF2Excel that implements the interface IPDF2Excel. The interface consists of the following methods.

### **PDF2Excel**

#### ***Prototype***

```
HRESULT PDF2Excel([in] BSTR sourceFile, [in] BSTR destFile,  
[out,retval] VARIANT_BOOL *successFlag)
```

#### ***Description***

The PDF2Excel() method converts the user specified file from PDF to Excel format. The output file may be opened with any version of Microsoft Excel. The original PDF file is not modified.

#### ***Parameters***

- |                    |  |
|--------------------|--|
| <b>sourceFile</b>  | Unicode string containing the absolute or relative filename of the PDF file that will be converted to Excel format.                    |
| <b>destFile</b>    | Unicode string containing the absolute or relative filename of the Excel file that contains the result of conversion.                  |
| <b>SuccessFlag</b> | Pointer to boolean variable that will contain the result of the operation: true is the conversion operation is successful; else false. |

#### ***Returns***

HRESULT enumerated value indicating the result of the COM operation.

#### ***Example***

```
HRESULT result = PDF2Excel(BSTR_sourceFile BSTR_destinationFile, &boolFlag);
```

## PDF2CSV

### **Prototype**

```
HRESULT PDF2CSV([in] BSTR sourceFile, [in] BSTR destFile,  
[out,retval] VARIANT_BOOL *successFlag)
```

### **Description**

The PDF2CSV() method converts the user specified file from PDF to Comma Separated Value format. The output file may be opened with a text editor or any version of Microsoft Excel. The original PDF file is not modified.

### **Parameters**

- |                   |  |
|-------------------|--|
| <b>sourceFile</b> | Unicode string containing the absolute or relative filename of the PDF file that will be converted to CSV format.                      |
| <b>destFile</b>   | Unicode string containing the absolute or relative filename of the CSV file that contains the result of conversion.                    |
| <b>success</b>    | Pointer to boolean variable that will contain the result of the operation: true is the conversion operation is successful; else false. |

### **Returns**

HRESULT enumerated value indicating the result of the COM operation.

### **Example**

```
HRESULT result = PDF2CSV(BSTR_sourceFile BSTR_destinationFile, &boolFlag);
```

## TemplatePDF2Excel

### **Prototype**

```
HRESULT TemplatePDF2Excel([in] BSTR sourceFile, [in] BSTR  
destFile, [in] BSTR templateFile, [out,retval] VARIANT_BOOL  
*successFlag)
```

### **Description**

The TemplatePDF2Excel() method converts the user specified file from PDF to Excel format. It uses the column positions described in the template file to convert data on the PDF file. The

output file may be opened with any version of Microsoft Excel. The original PDF file is not modified.

### **Parameters**

<b>sourceFile</b>	Unicode string containing the absolute or relative filename of the PDF file that will be converted to Excel format.
<b>destFile</b>	Unicode string containing the absolute or relative filename of the Excel file that contains the result of conversion.
<b>templateFile</b>	Unicode string containing the absolute or relative filename of the template file that describes the position of columns in a table of data to be converted.
<b>success</b>	Pointer to boolean variable that will contain the result of the operation: true is the conversion operation is successful; else false.

### **Returns**

HRESULT enumerated value indicating the result of the COM operation.

### **Example**

```
HRESULT result = TemplatePDF2Excel(BSTR_sourceFile BSTR_destinationFile,
BSTR_templateFile, &boolFlag);
```

## **TemplatePDF2CSV**

### **Prototype**

```
HRESULT TemplatePDF2CSV([in] BSTR sourceFile, [in] BSTR
destFile, [in] BSTR templateFile, [out,retval] VARIANT_BOOL
*sucessFlag)
```

### **Description**

The TemplatePDF2CSV() method converts the user specified file from PDF to Comma Separated Value format. It uses the column positions described in the template file to convert data on the PDF file. The output file may be opened with a text editor or any version of Microsoft Excel. The original PDF file is not modified.

### **Parameters**

- sourceFile** Unicode string containing the absolute or relative filename of the PDF file that will be converted to CSV format.
- destFile** Unicode string containing the absolute or relative filename of the CSV file that contains the result of conversion.
- templateFile** Unicode string containing the absolute or relative filename of the template file that describes the position of columns in a table of data to be converted.
- Success** Pointer to boolean variable that will contain the result of the operation: true is the conversion operation is successful; else false.

### **Returns**

HRESULT enumerated value indicating the result of the COM operation.

### **Example**

```
HRESULT result = TemplatePDF2CSV(BSTR_sourceFile, BSTR_destinationFile,  
BSTR_templateFile, &boolFlag);
```

# Index

A	
Able2Extract Command Line	
Installation Instructions	4
System Requirements	4
C	
Conversion DLL	3
Customer Service	2
I	
Investintech.com Inc.	1
InvestintechSDK DLL	4
Implicit Linking	11
InvestintechSDK_IDString	45
InvestintechSDK_PDF_to_Excel21	15, 16, 17, 29, 30, 31, 32, 33, 34, 44
Methods	13, 46
InvestintechSDK.dll	11
InvestintechSDK.h	11
InvestintechSDK.lib	11
T	
Technical Support	2





ERROR: undefined  
OFFENDING COMMAND: ceo

STACK:

/Ref\_part3  
/Dest  
-mark-