Users Guide For

PDFConverterX

By SoftInterface, Inc.

Version 1.X
Windows 95/98/2000/XP

Contents

PDFConverterX Users Manual

What is PDFConverterX?

PDFConverterX (PDFCX) is an ActiveX component/DLL designed to assist you, the developer, to quickly add a PDF conversion utility to your application. A tool like this can be very helpful when the native binary format of PDF documents is not acceptable. This component can convert a PDF to a Text file, or HTML file.

Please Note: This product does not require any other product to function. It is completely standalone.

PDFCX encapsulates all the details required for quick integration. Furthermore, a sample Visual Basic application is provided to get you up to speed quickly. Although you may not be using Visual Basic, the approach for all development environments will be similar.

Simply install the product, and add the component/DLL to your development environment. Once in place the routines can be accessed to programmatically convert PDF files. You will see that the code required to integrate this component in an application is relatively small considering the functionality it provides.

This document also discusses the Properties, Methods, and Events that are exposed by the PDFCX component. Please check Distributing PDFCX for information concerning which files are required when distributing the PDFCX.

At SoftInterface, Inc. we are constantly enhancing and improving our products. Please visit our web site to see what's new and tell us what you would like to see in our products (WWW.SoftInterface.COM). ***Also, it is important to register your products to ensure you have the latest version and support.***

PDFConverterX Features

- Quick integration (Sample Source code provided)
- Does not require Adobe Acrobat, it is completely standalone technology.
- This component can be invisible at run time.
- File conversion is exceptionally fast.
- DLL and ActiveX and EXE interfaces are available. For command line execution see 'Convert Doc'.

What is an ActiveX control?

ActiveX controls are an extension of Microsoft's COM (Component Object Model) technology, providing unprecedented compatibility with almost any rapid application development environment. ActiveX controls, sometimes referred to as *reusable components*, give you, the programmer, the easiest way to incorporate advanced functionality into your applications with little or no programming.

What is a Dynamic Link Library

A Dynamic link Library (DLL) is a standardized file format used to store algorithms. Because it is standardized, most Windows development environments can access the routines. A long, long time ago, when there was an operating system called DOS, 'static' link libraries existed. Static link libraries were bound to the final executable at link time. DLL's in contrast, remain as separate modules, allowing for the usage of the same file by multiple executables. This was critical during the initial Windows days, as hard disk space was minuscule by today's standards. DLL's typically have a file extension of *.DLL or *.OCX.

What You Will Need To Use PDFConverterX

The minimum hardware and software requirements to install and support the use of PDFCX are:

- IBM or compatible PC/AT (Pentium or higher CPU) with 16 MB of memory and one hard disk drive with 3 MB of space
- VGA or SVGA display adapter
- Microsoft Windows 95, Windows 98, Windows 2000, Windows XP.
- Development environment supporting 32-bit OCX controls such as Microsoft's Visual Basic (4.x or greater).

PDFConverterX Demo Source

A sample VB program is provided to illustrate proper usage of this PDFCX. Converting a PDF file is as simple as filling out a few properties, and then calling the ConvertPDFToFile() method. A sample piece of VB Code is shown below:

```
Public Function ConvertPDFToFile(sInputFile As String, _
                            sOutputFile As String, _
                            bIgnoreProtect As Boolean, _
                            sPasswordUser As String, _
                            sPasswordOwner As String, _
                            lFirstPage As Long, _
                            lLastPage As Long, _
                            lTargetFileType As Long, _
                            sErr As String) As Long
  Dim lResult As Long
  Dim bSingleShot As Boolean
  Dim sPassword As String

  bSingleShot = False
  sPassword = ""
```

```
  Set PDFX1= CreateObject("PDFConverterX.PDFX")
  PDFX1.Key = "0" ' Set to valid key upon purchase

  PDFX1.InputFile = sInputFile
  PDFX1.OutputFile = sOutputFile
  PDFX1.PasswordOwner = sPasswordUser
  PDFX1.PasswordUser = sPasswordOwner
  PDFX1.TargetFileType = lTargetFileType
  PDFX1.FirstPage = lFirstPage
  PDFX1.LastPage = lLastPage

  ConvertPDFToFile = PDFX1.ConvertPDFToFile()
  sErr = PDFX1.ErrorString
  set PDFFX1 = Nothing
End Function
```

Installing

When you run the setup program to install PDFCX on your computer, you will be able to specify where on your hard drive to install. It is preferred that you install it in the suggested directory for consistency (although not required).

Run the Setup.EXE that came with the PDFCX media. You may do this by clicking the *Start* button from the taskbar and select the *Run...* menu option. Then type the path and location of the Setup.EXE program. For example:

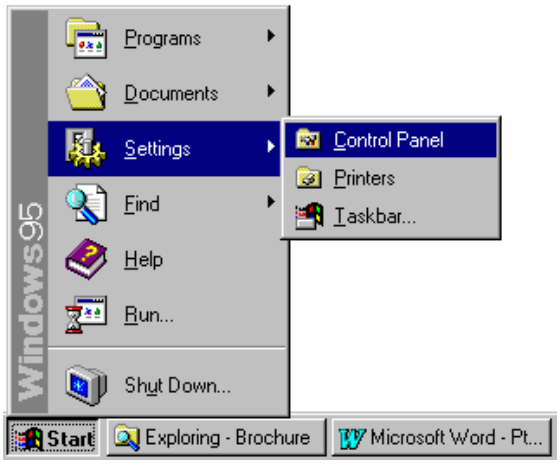
```
A:setup
```

Then press ENTER. Follow the installation instructions on the screen.

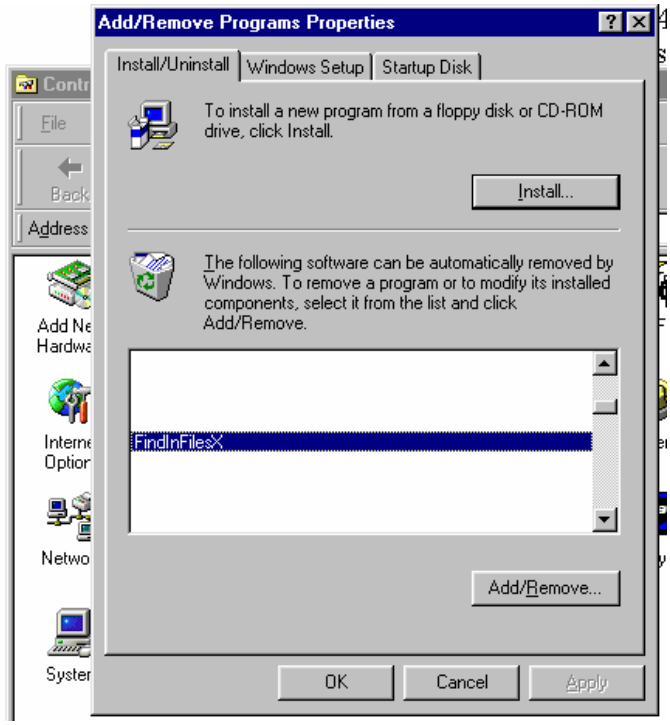
Uninstalling

It is highly suggested that you uninstall PDFCX before upgrading to a newer version of the product.

To uninstall PDFCX click the *Start* button from the taskbar and select *Settings* then *Control Panel*.



Within the control panel select the *Add/Remove Programs* icon. Double click on the PDFCX entry in the list box or push the *Add/Remove* button to uninstall.



All files copied during the installation will be removed (only if other programs are not currently dependent on them). Furthermore, if files have been added to the installation directory (i.e. program files you created) then the uninstall wizard will report that not all directories could be deleted. You will have to manually remove these files.

Distributing PDFConverterX

The necessary files needed to deploy your applications with PDFCX are discussed here. PDFCX is a self-registering ActiveX component.

The following table lists dependencies of PDFCX (**Items in bold were shipped with PDFConverterX.OCX**):

PDFCONVERTERX IMMEDIATE DEPENDENCIES	THIS DLL ALSO DEPENDS ON	VERSION IN USE AT TIME OF SHIPPING
SII_PDF.DLL	Kernel32.DLL, MSVCRT.DLL	Dated 5/6/2003
Comdlg32.OCX (Self Registering)	USER32.DLL, GDI32.DLL, KERNEL32.DLL, ADVAPI32.DLL, OLE32.DLL, COMDLG32.DLL	6.0.84.18
MSVCRT.DLL	Kernel32.DLL	6.1.8637.0
KERNEL32.DLL		4.10.1998
USER32.DLL	ADVAPI32.DLL KERNEL32.DLL, GDI32.DLL	4.10.1998
GDI32.DLL	KERNEL32.DLL, ADVAPI32.DLL	4.10.1998

Dependency implies that the file must be installed before PDFConverterX.OCX is installed. Do not distribute (of course) Kernel32.DLL, USER32.DLL, and GDI32.DLL as these are operating system dependent.

Troubleshooting PDFConverterX - Updates

The SoftInterface, Inc. web site (www.SoftInterface.COM) will have the links required for all PDFCX:

- Appendix A - Solutions to Common Issues
- Frequently Asked Questions
- Bug Lists
- Latest Patches/Downloads

Please first review this manual, then the SoftInterface, Inc. Web site for assistance.

Finally, if you still have trouble you may e-mail for support at Support@SoftInterface.COM.

PDFConverterX Reference

Reference Introduction

Here we discuss the Properties, Methods and Events (attributes) of PDFCX. The properties have been separated in two sections for ease of use. "Properties – Standard" discusses the properties most often used with the **ConvertPDFToFile()** method. "Properties – Text Settings" goes over the many items that can be set to fine tune the conversion process of textual content for *special* situations only.

Properties - Standard

FirstPage, LastPage

Data Type: Long

Default Value: 0

Description:

FirstPage: *First page to begin converting. If the value is less than 1 then it is defaulted to the first page of the PDF.*

LastPage: *Last page to convert. If the value is less than 1 then it is defaulted to the last page of the PDF.*

Set both FirstPage and LastPage to 0 to do the whole document.

Example:

```
Public Function ConvertPDFToFile(sInputFile As String, _
                            sOutputFile As String, _
                            bIgnoreProtect As Boolean, _
                            sPasswordUser As String, _
                            sPasswordOwner As String, _
                            lFirstPage As Long, _
                            lLastPage As Long, _
                            lTargetFileType As Long, _
                            sErr As String) As Long
  Dim lResult As Long
  Dim bSingleShot As Boolean
  Dim sPassword As String

  bSingleShot = False
  sPassword = ""

  PDFX1.InputFile = sInputFile
```

```
  PDFX1.OutputFile = sOutputFile
  PDFX1.PasswordOwner = sPasswordUser
  PDFX1.PasswordUser = sPasswordOwner
  PDFX1.TargetFileType = lTargetFileType
  PDFX1.FirstPage = lFirstPage
  PDFX1.LastPage = lLastPage

  ConvertPDFToFile = PDFX1.ConvertPDFToFile()
  sErr = PDFX1.ErrorString
End Function
```

IgnoreProtect

Data Type: Boolean

Default Value: FALSE

Description: The property, if set to TRUE, can sometimes be used to override user password protection of the PDF file. Set this property to FALSE to enforce password protection.

InputFile

Data Type: String

Default Value: Empty

Description: Set this property to the path of the file you want to convert.

See also: OutputFile,

Example:

```
Public Function ConvertPDFToFile(sInputFile As String, _
                            sOutputFile As String, _
                            bIgnoreProtect As Boolean, _
                            sPasswordUser As String, _
                            sPasswordOwner As String, _
                            lFirstPage As Long, _
                            lLastPage As Long, _
                            lTargetFileType As Long, _
                            sErr As String) As Long
  Dim lResult As Long
  Dim bSingleShot As Boolean
  Dim sPassword As String

  bSingleShot = False
  sPassword = ""

  PDFX1.InputFile = sInputFile
  PDFX1.OutputFile = sOutputFile
  PDFX1.PasswordOwner = sPasswordUser
  PDFX1.PasswordUser = sPasswordOwner
  PDFX1.TargetFileType = lTargetFileType
  PDFX1.FirstPage = lFirstPage
  PDFX1.LastPage = lLastPage

  ConvertPDFToFile = PDFX1.ConvertPDFToFile()
  sErr = PDFX1.ErrorString
End Function
```

Key

Data Type: String

Default Value: 0

Description: After purchase you will receive a reseller key. Set the Key property to this value.

OutputFile

Data Type: String

Default Value: Empty

Description: Set this property to the path of the file you want to convert the file to.

See also: InputFile

Example:

```
Public Function ConvertPDFToFile(sInputFile As String, _
                            sOutputFile As String, _
                            bIgnoreProtect As Boolean, _
                            sPasswordUser As String, _
                            sPasswordOwner As String, _
                            lFirstPage As Long, _
                            lLastPage As Long, _
                            lTargetFileType As Long, _
                            sErr As String) As Long
  Dim lResult As Long
  Dim bSingleShot As Boolean
  Dim sPassword As String

  bSingleShot = False
  sPassword = ""

  PDFX1.InputFile = sInputFile
  PDFX1.OutputFile = sOutputFile
  PDFX1.PasswordOwner = sPasswordUser
  PDFX1.PasswordUser = sPasswordOwner
  PDFX1.TargetFileType = lTargetFileType
  PDFX1.FirstPage = lFirstPage
  PDFX1.LastPage = lLastPage

  ConvertPDFToFile = PDFX1.ConvertPDFToFile()
  sErr = PDFX1.ErrorString
End Function
```

PasswordOwner, PasswordUser

Data Type: String

Default Value: Empty

Description: Set these properties to the passwords that may be required. PasswordOwner is specified by the creator of the PDF file. PasswordUser is specified by a user of the PDF file.

Example:

```
Public Function ConvertPDFToFile(sInputFile As String, _
                            sOutputFile As String, _
                            bIgnoreProtect As Boolean, _
                            sPasswordUser As String, _
                            sPasswordOwner As String, _
                            lFirstPage As Long, _
                            lLastPage As Long, _
                            lTargetFileType As Long, _
                            sErr As String) As Long
  Dim lResult As Long
  Dim bSingleShot As Boolean
  Dim sPassword As String

  bSingleShot = False
```

```
  sPassword = ""

  PDFX1.InputFile = sInputFile
  PDFX1.OutputFile = sOutputFile
  PDFX1.PasswordOwner = sPasswordUser
  PDFX1.PasswordUser = sPasswordOwner
  PDFX1.TargetFileType = lTargetFileType
  PDFX1.FirstPage = lFirstPage
  PDFX1.LastPage = lLastPage

  ConvertPDFToFile = PDFX1.ConvertPDFToFile()
  sErr = PDFX1.ErrorString
End Function
```

StateFileUser

Data Type: String

Default Value: WindowsSystemFolder\PDFCX.INI

Description: All the properties (except StateFileUser of course) associated with this component are saved to an INI file. You can specify which INI file to use. StateFileUser is always saved to the default file (WindowsSystemFolder\PDFCX.INI).

See also: SaveSettingsUser(), LoadSettingsUser()

Example:

```
Private Function LoadSettingsFile() As Boolean
  If (FileExists(cbStateFileUser.Text) = False) Then
    ' Handle error
    Call MsgBox("Settings File does not exist: " +
cbStateFileUser.Text, vbOKOnly)
    LoadSettingsFile = False
    Exit Function
  End If

  pdfcx.StateFileUser = cbStateFileUser.Text ' Make sure we have the
selected User State file

  If (pdfcx.LoadSettingsUser = False) Then
    LoadSettingsFile = False
    Exit Function
  End If
End Function
```

TargetFileType

Data Type: Long

Default Value: 1 (Text File)

Description: Target file type. Either 1 for Text or 2 for HTML.

Example:

```
Public Function ConvertPDFToFile(sInputFile As String, _
                            sOutputFile As String, _
                            bIgnoreProtect As Boolean, _
                            sPasswordUser As String, _
                            sPasswordOwner As String, _
                            lFirstPage As Long, _
                            lLastPage As Long, _
                            lTargetFileType As Long, _
                            sErr As String) As Long
  Dim lResult As Long
  Dim bSingleShot As Boolean
  Dim sPassword As String
```

```
  bSingleShot = False
  sPassword = ""

  PDFX1.InputFile = sInputFile
  PDFX1.OutputFile = sOutputFile
  PDFX1.PasswordOwner = sPasswordUser
  PDFX1.PasswordUser = sPasswordOwner
  PDFX1.TargetFileType = lTargetFileType
  PDFX1.FirstPage = lFirstPage
  PDFX1.LastPage = lLastPage

  ConvertPDFToFile = PDFX1.ConvertPDFToFile()
  sErr = PDFX1.ErrorString
End Function
```

Properties – Text Settings

These functions are used to adjust the text output for special situations. (see Appendix A and Appendix B). All properties are prefixed with "ts" which is short for Text Settings.

tsWordMinSpaceWidth, tsWordMaxSpaceWidth

Data Type: double

Default Value: tsWordMinSpaceWidth = 0.15, tsWordMaxSpaceWidth = 2.0

Description: Minimum and maximum inter-word spacing (as a fraction of the average character width).

This is the smallest and largest space between 2 letters to make it the same word.

If two words are split and should NOT be, adjust the max.

If two words are not split and they SHOULD be – adjust the min.

tsWordDefMinSpaceWidth, tsWordDefMaxSpaceWidth

Data Type: double

Default Value: tsWordDefMinSpaceWidth= 0.09, tsWordDefMaxSpaceWidth= 1.5

Description: Default min and max inter-word spacing (when the average character width is unknown – if there isn't a font definition for the font).

These are the default spacing to use if NO EMBEDDED FONT exists in the PDF file. Normally this is not used (most PDF files have embedded font details).

tsSetDupMaxDeltaX, tsSetDupMaxDeltaY

Data Type: double

Default Value: tsSetDupMaxDeltaX= 0.4, tsSetDupMaxDeltaY= 0.4

Description:

Maximum difference in x, y coordinates (as a fraction of the font size) allowed for duplicated text (fake boldface, drop shadows), which is to be discarded. See Appendix B, Bold Text for further explanation.

If duplicate text is in the output as a result of "fake" boldface, try decreasing these numbers until the duplicate text is removed.

If VALID duplicates are being erroneously removed, increase the values here to restore the text to the output.

Example Text: "Hello World! Why are letters dropped?"

Output: "Helo World! Why are leters droped?"

Solution: DECREASE the tsSetDupMaxDeltaX 1/10th at a time.

The default is 0.4 - try 0.3 then 0.2 then 0.1

This is the distance between 2 chars to be considered as VALID text.

So right now, the duplicate letters are CLOSER then the 0.4 ratio - so it is TOSSING the second.

tsLineOverlapSlack

Data Type: double

Default Value: tsLineOverlapSlack= 0.3

Description:

Min overlap (as a fraction of the font size) required for two lines to be considered vertically overlapping.

If 2 lines overlap on the PDF, they must either be merged onto the SAME line in the output OR moved to separate lines.

Increase this number if output is desired to be on the SAME line, decrease this value to make sure output is on separate lines.

tsLineMaxBaselineDelta

Data Type: double

Default Value: tsLineMaxBaselineDelta= 0.1

Description:

Max difference in baseline y coordinates (as a fraction of the font size) allowed for words, which are to be grouped into a line, not including sub/superscripts.

Similar to the overlap, this is used to group a "sentence" together all on the same line

tsLineMaxFontSizeRatio

Data Type: double

Default Value: tsMaxFontSizeRatio = 1.4

Description:

Max ratio of font sizes allowed for words, which are to be grouped into a line, not including sub/superscripts.

If multiple fonts are used within the SAME line, we either need to keep all text on the same output line, or potentially move the different font portion to its own line, depending on the size. Increase this number for text output to be on the same line, decrease it to force different fonts to be on separate lines.

tsLineMinDeltaX

Data Type: double

Default Value: tsLineMinDeltaX = -0.5

Description:

Min spacing (as a fraction of the font size) allowed between words, which are to be grouped into a line.

tsLineMinSubscriptOverlap, tsLineMinSuperscriptOverlap

Data Type: double

Default Value: tsLineMinSubscriptOverlap = 0.3, tsLineMinSuperscriptOverlap = 0.3

Description:

Minimum vertical overlap (as a fraction of the font size) required for superscript and subscript words.

tsLineMinSubscriptFontSizeRatio, tsLineMaxSubscriptFontSizeRatio, tsLineMinSuperscriptFontSizeRatio, tsLineMaxSuperscriptFontSizeRatio

Data Type: double

Default Value: tsLineMinSubscriptFontSizeRatio= 0.4, tsLineMaxSubscriptFontSizeRatio= 1.01, tsLineMinSuperscriptFontSizeRatio= 0.4, tsLineMaxSuperscriptFontSizeRatio= 1.01

Description:

Minimum vertical overlap (as a fraction of the font size) required for superscript and subscript words.

Min/max ratio of font sizes allowed for sub/superscripts compared to the base text.

If superscript and subscript are used within the PDF and are based on FONT SIZES within the same text line (x,y), changing the min and max will adjust whether or not the text is on the same line or a new line as the existing text it is super or subbing.

tsLineMaxSubscriptDeltaX, tsLineMaxSuperscriptDeltaX

Data Type: double

Default Value: tsLineMaxSubscriptDeltaX = 0.2, tsLineMaxSuperscriptDeltaX= 0.2

Description:

Max horizontal spacing (as a fraction of the font size) allowed before sub/superscripts.

If super/sub scripts are present as the same font, but on a different row, adjusting these values will allow you to either merge the super/sub to the same row as the text, or make it a different row.

tsBlockMaxSpacing

Data Type: double

Default Value: tsBlockMaxSpacing= 2.0

Description:

Maximum vertical spacing (as a fraction of the font size) allowed for lines, which are to be grouped into a block.

This allows lines to be grouped so columns can line up properly. If a "grid" is displayed for instance and each row is far apart (due to whitespace desired in the PDF or some other reason) then you can increase this value to ensure that the columns will properly line up.

If the output mistakenly assumes the lines are together, when in fact they are NOT and the discrepancy in the column in desired – then DECREASE this value to ensure the rows are NOT grouped together.

tsBlockMaxFontSizeRatio

Data Type: double

Default Value: tsBlockMaxFontSizeRatio= 1.3

Description:

Max ratio of primary font sizes allowed for lines, which are to be grouped into a block.

Similar to the blkMaxSpacing – this is for grouping different rows with different FONTS. This is so different fonts will still line up in the output file if they are part of the same "column", unless the font is DRASTICALLY different in size. – which is where this property comes in.

tsBlockOverlapSlack

Data Type: double

Default Value: tsBlockkOverlapSlack = 0.5

Description:

Min overlap (as a fraction of the font size) required for two blocks to be considered vertically overlapping.

tsFlowMaxDeltaY

Data Type: double

Default Value: tsFlowMaxDeltaY= 1.0

Description:

Max vertical offset (as a fraction of the font size) of the top and bottom edges allowed for blocks, which are to be grouped into a flow.

Methods

ConvertPDFToFile()

Description: This method is used to display a built in dialog box (shown below) for quick modification of the many text setting parameters.

Parameters: The following properties are used by this function: **InputFile**, **OutputFile, PasswordOwner, PasswordUser, TargetFileType, FirstPage. LastPage**

Return Values (Long Integer):

Return Value	Meaning
0	Successfully converted the file
-1	Unable to open necessary SII DLL
-2	**InputFile** is invalid or could not open
-5	Couldn't read the page catalog
-6	PDF File is damaged
-7	File is encrypted and password was incorrect or not supplied
-8	User does not have permission to read
-9	Invalid last page range specified
-10	Shareware expired
-11	Conversion failed

See Also: InputFile, OutputFile, PasswordOwner, PasswordUser, TargetFileType, FirstPage. LastPage

Example:

```
Public Function ConvertPDFToFile(sInputFile As String, _
                                 sOutputFile As String, _
                                 bIgnoreProtect As Boolean, _
                                 sPasswordUser As String, _
                                 sPasswordOwner As String, _
                                 lFirstPage As Long, _
                                 lLastPage As Long, _
                                 lTargetFileType As Long, _
                                 sErr As String) As Long
  Dim lResult As Long
  Dim bSingleShot As Boolean
  Dim sPassword As String

  bSingleShot = False
  sPassword = ""

  PDFX1.InputFile = sInputFile
  PDFX1.OutputFile = sOutputFile
  PDFX1.PasswordOwner = sPasswordUser
  PDFX1.PasswordUser = sPasswordOwner
  PDFX1.TargetFileType = lTargetFileType
```

```
  PDFX1.FirstPage = lFirstPage
  PDFX1.LastPage = lLastPage

  ConvertPDFToFile = PDFX1.ConvertPDFToFile()
  sErr = PDFX1.ErrorString
End Function
```

LoadSettingsUser(), SaveSettingsUser()

Description: These methods are used to save and restore all properties of this component, except StateFileUser. If you want to save to a file other than the default (WindowsSystemFolder\PDFCX.INI) then you should set the StateFileUser property.

Parameters: None

Return Value: Boolean, TRUE if successful, otherwise FALSE.

See Also: **StateFileUser**

Example:

```
Private Function LoadSettingsFile() As Boolean
  If (FileExists(cbStateFileUser.Text) = False) Then
    ' Handle error
    Call MsgBox("Settings File does not exist: " +
cbStateFileUser.Text, vbOKOnly)
    LoadSettingsFile = False
    Exit Function
  End If

  pdfcx.StateFileUser = cbStateFileUser.Text ' Make sure we have the
selected User State file

  If (pdfcx.LoadSettingsUser = False) Then
    LoadSettingsFile = False
    Exit Function
  End If
End Function

Private Sub cmdApply_Click()
  pdfcx.StateFileUser = cbStateFileUser.Text ' Make sure we have the
selected User State file

  If (pdfcx.SaveSettingsUser = False) Then
    Call MsgBox("Unable to save to file, check that folder exists
and/or user rights.", vbOKOnly)
    Exit Sub
  End If
End Sub
```

RestoreDefaultTextSettings()

Description: This method resets all the Text Setting variables to their default values.

Parameters: None

Return Value: None

See Also: NA

ShowAbout(), ShowRegistration()

Description: These are for Softinterface use only.

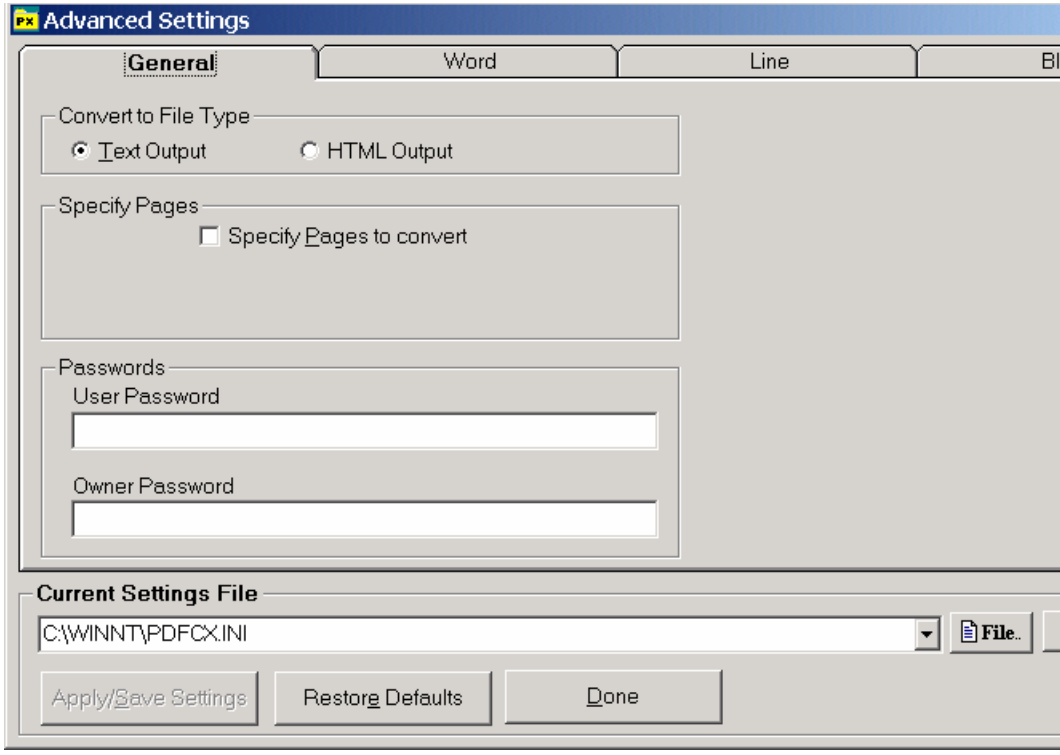
Parameters: None

Return Value: None

See Also: StateFileUser, LoadSettingsUser(), SaveSettingsUser()

ShowConvertPDFAdvancedDialog()

Description: This method is used to display a built in dialog box (shown below) for quick modification of the many text setting parameters.



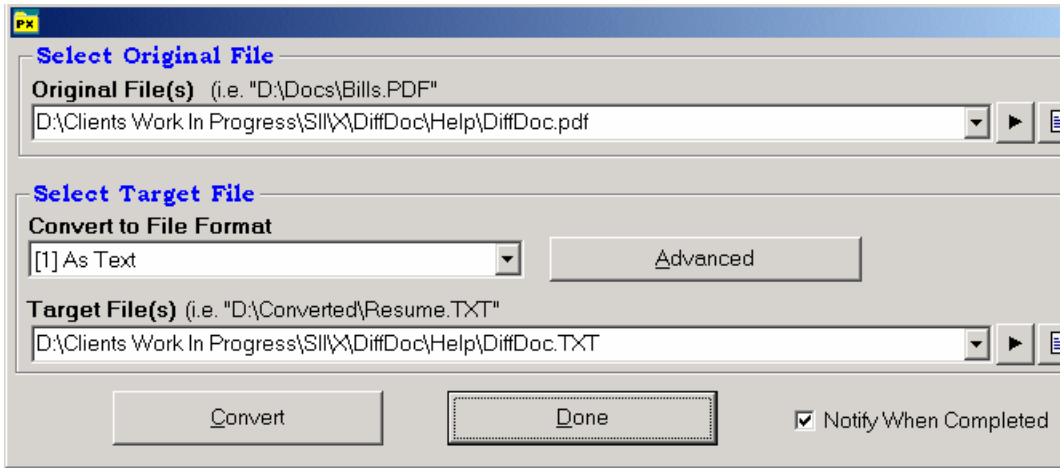
Parameters: None

Return Value: None

See Also: ShowConvertPDFDialog()

ShowConvertPDFDialog()

Description: This method is used to display a built in dialog box (shown below) for quick and easy file conversion tasks.



Parameters: None

Return Value: None

See Also: ShowConvertPDFAdvancedDialog()

Events

OnError(lErrorCode as long, sErrDescription As String)

The OnError event will fire whenever an error occurs within the component. Two parameters are passed, an error code (long integer) and an error string. Use this event to help troubleshoot any problems you may run into.

Working with the DLL Interface (To be completed)

The reference manual describes the methods and properties that can be accessed through the ActiveX interface. Exported DLL functions by the same name are within a file called SII_PDF.DLL. Although the function names are similar, there are some differences in the spelling of routines.

Use UnlockDLL() to initialize the library before calling the other functions. Pass the key provided, or 0 if not purchased yet.

The usage of the other functions is described in their respective method descriptions in this reference manual.

Access to the routines is different for each development environment. Visual Basic for example requires the following declarations in a module:

```
Public Declare Function pdfUnlockDLL Lib "SII_PDF.DLL" Alias "_UnlockDLL@4" (ByVal lKey As
Long) As Long

Public Declare Function pdfOpenPDF Lib "SII_PDF.DLL" Alias "_OpenPDF@20" (ByVal sFileName As
String, _
          ByVal bIgnoreProtect As Boolean, _
          ByVal sUserPassword As String, _
          ByVal sOwnerPassword As String, _
          ByRef lHandle As Long) As Long
```

```
Public Declare Function pdfClosePDF Lib "SII_PDF.DLL" Alias "_ClosePDF@4" (ByRef lHandle As
Long) As Long

Public Declare Function pdfGetNumPages Lib "SII_PDF.DLL" Alias "_GetNumPages@4" (ByRef lHandle
As Long) As Long
```

See the provided "modPDFConversion.BAS" module and sample visual basic program for these declarations.

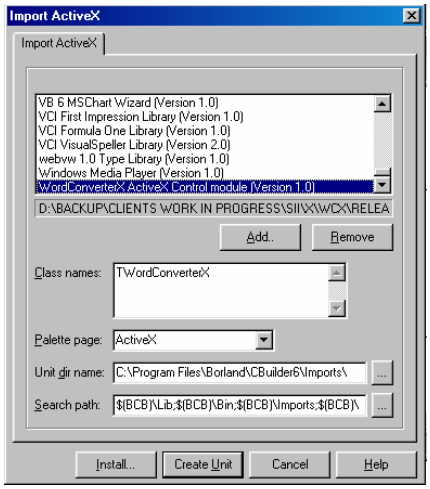
The functions in the following sections are available through the DLL interface.

Using Components In Borland C++ Builder

You will want to import the PDFConverterX components before using the demonstration source.

C++ Builder: Importing Components

After installation of this product, it can be made available from the 'ActiveX' tab of the C++ Builder controls palette by:



i. Select the **"Component\Import ActiveX Control…"** menu item.

ii. Select **PDFConverterX ActiveX Control Module** from the list of available components.

iii. Click the **Install** button to create a package. The name and location of the package is of your choosing.

iv. After this has completed the component palette will be updated as a result of rebuilding the package.

C++ Builder: Passing Arguments

All arguments to the methods or events can be viewed by the wrapper class, TPDFConverterX, created by the importing mechanism.

Passing string arguments that do not get modified by the calling method can be done by using the WideString data type as illustrated below:

```
WideString wsSourceFile,wsTargetFile;
wsSourceFile = txtWordSourceFile->Text;
```

```
  wsTargetFile = txtWordTargetFile->Text;
  bResult = PDFCX->ConvertWordDoc(wsSourceFile, wsTargetFile,
cWordType[cbWordConversionType->ItemIndex]);
```

Although none exist in TPDFConverterX, passing strings arguments that *get* modified by the calling method can done by using the WideString data type as illustrated below:

```
/*
VARIANT_BOOL __fastcall TCompareFilesX::DirCompareResultsGet(long
ResultIndex, BSTR* FileName, BSTR* DirMaster, BSTR* DirSource,long*
IsFile, long* OnlyInSource,long* OnlyInMaster, long*
SizeDifferent,long* DateDifferent,long* ContentDifferent)
*/
    WideString ws_file_name,ws_dir_mstr,ws_dir_src;
    if (CompareFilesX1->DirCompareResultsGet(ii, &ws_file_name,
&ws_dir_mstr, &ws_dir_src, &bIsFile, &bOnlyInSource, &bOnlyInMaster,
&bSizeDifferent, &bDateDifferent, &bContentDifferent) == FALSE)
      break;// Exit for loop

    AnsiString as_file_name = AnsiString(ws_file_name);
    AnsiString as_dir_mstr = AnsiString(ws_dir_mstr);
    AnsiString as_dir_src = AnsiString(ws_dir_src);
```

Appendix A – Solutions to Common Issues

Letters Are Dropped When In Duplicates

Example Text: "Hello World! Why are letters dropped?"

Output: "Helo World! Why are leters droped?"

Solution: DECREASE the tsSetDupMaxDeltaX 1/10th at a time.

The default is 0.4 - try 0.3 then 0.2 then 0.1

This is the distance between 2 chars to be considered as VALID text.

So right now, the duplicate letters are CLOSER then the 0.4 ratio - so it is TOSSING the second.

Appendix B – Output Formatting Issues

Unicode without embedded map

Unicode is supported as long as the map is included within the PDF. A default map is coded, so if Unicode map is NOT enclosed, it still MAY work, but could end up not matching exactly.

Font without embedded fonts

PDF creators usually embed the font type used within the PDF itself, but if a user explicitly selects to NOT embed the font, then issues can arise with the proper spacing. If a standard font is used, the DLL can recover and use the encodings provided within the code. If TRUETYPE fonts are used, there is no support for external font files, therefore a default will be used and may cause the spacing to be off.

Graphics

Graphics are obviously not supported within text extraction, but depending on how the text "interacts", a graphic with text surrounding it MAY cause the text to not be in the proper position. In MOST cases, the text is drawn within its own "text box" and graphics will not affect the text position.

Lines, curves and other drawing tools

These may change the position of surrounding text and could cause text to be moved to an undesired location in the pure text output.

Order of objects on the page

Text boxes that are "hidden" behind graphics or other text boxes in the PDF will still be extracted and shown in the text output. There is no provision for the ORDER of text objects – ALL text objects will be processed.

Color of text on page

Color of text is ignored. If a text section is made the same color as the background to remove it instead of removing the text box itself – this text will still be in the output.

Superscript/Subscript

Superscript and subscript fonts may end up appearing either on the same line, or on lines by themselves – depending on the size and the settings provided.

Misshapen text boxes

Text boxes that are distorted, backwards (pulled the wrong way when creating the PDF) etc. – WILL be processed. In the majority of cases – this text is most likely "hidden" behind a graphic and forgotten – but will show up in the ACSII processing. If "extra" characters seem to appear in the output that don't show in the PDF viewer, then this most likely is the case.

Bold text

Some PDF creators duplicate text at slightly off x,y coordinates – basically copying the text to be bold and moving the duplicate a little lower and a little to the right – resulting in the same text twice on the screen at different spacing – creating the bold effect. When we extract text – this can result in text being duplicated. Settings have been provided to adjust for this.

Fonts and positioning

Certain fonts mixed with other fonts COULD cause issues when trying to maintain the spacing/order of the original PDF. The best possible format is maintained.

Tiny fonts

Fonts within a PDF that are LESS THAN 3 points are ignored.

Annotations

Annotations are not currently supported and are ignored.

Graphic "paths"

Graphic paths (embedded graphics) are not supported and are ignored.

Links

Links are not followed and not processed, however the text itself (depending on how the link was created) should still be maintained in the output.

Software License and Software Disclaimer of Warranty

License

This a legal document which is an agreement between you, the Licensee, and SoftInterface, Inc.. By opening/installing the software, Licensee agrees to become bound by the terms of this agreement, which include the Software License and Software Disclaimer of Warranty.

This software, (hereinafter "Software") is owned by SoftInterface, Inc., which reserves all rights not granted to you by this Agreement. This Software is copyrighted and protected by both the United States copyright laws and international treaty provisions. As the purchaser of this Software, you receive only ownership of the copyrighted written material and program disks in the package and a non-exclusive license to use the Software recorded on the program disks, on the following terms and conditions.

You May:

1. Use the Software on only *one computer* by one user at a time, even if the Software is distributed in more than one disk size or format. If you desire additional users of the Software, such as on a network or multi-user computer, you must purchase an additional copy of the Software for each simultaneous user. The Software is in "use" on a computer when it is loaded into temporary memory (i.e. RAM) or installed into permanent memory (e.g. hard disk, or other storage device) of that computer.
2. Make one backup copy of the software. The Software is owned by SoftInterface, Inc. or its suppliers and is protected by United States copyright laws and international treaty provisions. Therefore, you must treat the Software like any other copyrighted material (e.g. a book or a musical recording) except that you may either (a) make one copy of the Software solely for backup or archival purposes, or (b) transfer the Software to a single hard disk provided you keep the original solely for backup or archival purposes. You may not make multiple copies of Software nor the written materials accompanying the Software.
3. Transfer the Software and accompanying written materials on a permanent basis provided you retain no copies and the recipient agrees to the terms of this Agreement.
4. Copy, alter and modify all example source code included in the Software with the exception than any SoftInterface, Inc.'s copyright notices may not be removed or altered.
5. Use the Software to produce one standalone program "Application Software Program" which is the executable files to accomplish certain tasks. You have a royalty-free right to distribute only the "run-time modules" with the executable files created in any other vendor product (Language or development tool) limited as hereinafter set forth in paragraph a through d. SoftInterface, Inc. grants you a royalty-free distribution if: (a) you distribute the "run time" modules only in conjunction with the executable files that make use of them as a part of your software product; (b) you do not use the SoftInterface, Inc. name, logo or trademark to market your software product; (c) The Software end users do not use the "run time" modules or any other Software components for development purposes. and, (d) you agree to indemnify, hold harmless, and defend SoftInterface, Inc. and its suppliers from and against any and all claims or lawsuits including attorney's fees, that arise or result from the use or distribution of your software product. If any of the conditions set forth in paragraph a through d are breached, such breach shall constitute an unlawful use of the Software, and you shall be prosecuted to the full extent of the law. Furthermore, you shall be liable to SoftInterface, Inc. for all damages caused by such breach and unlawful use of the Software, including attorney's fees and costs incurred in any action, lawsuit or claim brought or filed to redress the breach of this agreement. The "run time modules" are those files included in the Software package that are required during execution of your software program.

Redistribution Rights

SoftInterface, Inc. grants you a license to distribute the Application Software Program you create with the following terms and conditions:

1. You must not permit further redistribution of the Redistributable Files by your end-user customers.
2. The Application Software Program is distributed in executable form only, and no SoftInterface, Inc.'s source code, ASCII format files are distributed.
3. The Application Software Program contains valid copyright notices and is licensed to the end user pursuant to an appropriate license agreement.
4. SoftInterface, Inc.'s copyright notices must not be removed from any SoftInterface, Inc.'s Software included in the Application Software Program.
5. SoftInterface, Inc.'s has no warranty obligation to any user of the Application Software Program.
6. SoftInterface, Inc. has no obligation to provide technical support, services, upgrades or other support to any user of Application Software Program.

7. The Application Software Program represents value added by you, and is not just a repackaging of SoftInterface, Inc.'s Software.
8. You indemnify and hold SoftInterface, Inc. harmless from any claims arising out of the use, sale, license or other transfer of the Application Software Program.

Restrictions
You May Not:
1. Wrap this OCX into another OCX or component based software.
2. Use or permit this Software to be used except as permitted by this Agreement.
3. Loan, rent, copy or sub-license this Software except as permitted above.
4. Alter, modify, reverse engineer, decompose or disassemble compiled, binary format files included in the Software.
5. Export this product in violation of the Export Administration Act and the regulations thereunder.
6. Duplicate the written documentation accompanying the Software.

Term
This License is effective until terminated. This License will terminate automatically without notice from SoftInterface, Inc. if Licensee fails to comply with any term or condition of this License. The Licensee agrees upon such termination to return or destroy the written materials and all copies of the software. The Licensee may terminate the agreement by returning or destroying the program and documentation and all copies thereof.

Limited Warranty

SoftInterface, Inc. warrants that the Software will perform substantially in accordance with the written materials and the program disk, instructional manuals and reference materials are free from defects in materials and workmanship under normal use for 90 days from the date of receipt. All express or implied warranties of the software and related materials are limited to 90 days.

Except as specifically set forth herein, the Software and accompanying written materials (including instructions for use) are provided "as is" without warranty of any kind. Further, SoftInterface, Inc. does not warrant, guarantee, or make any representations regarding the use, or the results of the use, of the software or written materials in terms of correctness, accuracy, reliability, currentness, or otherwise. The entire risk as to the results and performance of the software is assumed by Licensee and not by SoftInterface, Inc. or its distributors, agents or employees.

EXCEPT AS SET FORTH HEREIN, THERE ARE NO OTHER WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WITH RESPECT TO THE SOFTWARE, THE ACCOMPANYING WRITTEN MATERIALS, AND ANY ACCOMPANYING HARDWARE.

Remedy:
SoftInterface, Inc.'s entire liability and the Licensee's exclusive remedy shall be, at SoftInterface, Inc.'s option, either (a) return of the price paid or (b) repair or replacement of the software or accompanying materials. In the event of a defect in material or workmanship, the item may be returned within the warranty period to SoftInterface, Inc. for a replacement without charge, provided the license previously sent to the limited warranty registration card to SoftInterface, Inc., or can furnish proof of the purchase of the program. This remedy is void if failure has resulted from accident, abuse, or misapplication. Any replacement will be warranted for the remainder of the original warranty period.

NEITHER SOFTINTERFACE, INC. NOR ANYONE ELSE WHO HAS BEEN INVOLVED IN THE CREATION, PRODUCTION, SALE OR DELIVERY OF THIS PRODUCT SHALL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, OR INCIDENTAL DAMAGES (INCLUDING DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION AND THE LIKE) ARISING OUT OF THE USE OF OR THE INABILITY TO USE SUCH PRODUCT EVEN IF SOFTINTERFACE, INC. HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, BECAUSE SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF LIABILTY FOR CONSEQUENTIAL OR INCIDENTAL DAMAGES, OR LIMITATIONS ON DURATION OF IMPLIED WARRANTY, THE ABOVE LIMITATIONS MAY NOT APPLY TO LICENSEE.

U.S. Government Rights
This Software was developed entirely with private funds, and not as part of the performance of a contract with the U.S. Government. The U.S. Government has no special rights with respect to use, duplication or disclosure, other than as provided for in the Agreement.

General
Should any provision of this Agreement be void or not enforceable, the remainder of this Agreement shall continue in full force and effect.

This agreement is governed by the laws of the state of California.