

**Foxit<sup>®</sup>**



**User Manual**  
**Foxit<sup>®</sup> PDF SDK (ActiveX 4.0)**  
**Programming Guide**  
*For Windows*



**Microsoft<sup>®</sup>**  
**GOLD CERTIFIED**  
Partner

©2011 Foxit Corporation. All rights reserved.

Copyright © 2011 Foxit Corporation. All Rights Reserved.

No part of this document can be reproduced, transferred, distributed or stored in any format without the prior written permission of Foxit.

Anti-Grain Geometry - Version 2.3, Copyright (C) 2002-2005 Maxim Shemanarev (<http://www.antigrain.com>). FreeType2 (freetype2.2.1), Copyright (C) 1996-2001, 2002, 2003, 2004| David Turner, Robert Wilhelm, and Werner Lemberg. LibJPEG (jpeg V6b 27- Mar-1998), Copyright (C) 1991-1998 Independent JPEG Group. ZLib (zlib 1.2.2), Copyright (C) 1995-2003 Jean-loup Gailly and Mark Adler. Little CMS, Copyright (C) 1998-2004 Marti Maria. Kakadu, Copyright (C) 2001, David Taubman, The University of New South Wales (UNSW). PNG, Copyright (C) 1998-2009 Glenn Randers-Pehrson. LibTIFF, Copyright (C) 1988-1997 Sam Leffler and Copyright (C) 1991-1997 Silicon Graphics, Inc.

Permission to copy, use, modify, sell and distribute this software is granted provided this copyright notice appears in all copies. This software is provided "as is" without express or implied warranty, and with no claim as to its suitability for any purpose.

# Contents

<b>Overview .....</b>	<b>9</b>
<b>Tutorials .....</b>	<b>11</b>
1) Open a PDF File .....	11
2) Go to a specific page.....	11
3) Zoom a page .....	11
4) Rotate a page.....	11
5) Print a PDF document .....	12
6) Hide or show UI elements .....	12
7) Iterate the whole outline tree .....	12
8) Search a PDF document .....	12
9) *Annotations tools .....	12
10) #Form Application .....	12
<b>Foxit PDF SDK ActiveX control.....</b>	<b>14</b>
<b>Properties.....</b>	<b>14</b>
FilePath .....	14
Password.....	14
PageCount .....	14
CurPage .....	14
Rotate.....	14
Zoomlevel.....	15
CurrentTool.....	15
Printer.....	16
DocumentInfo .....	16
*bHasFormFields.....	17
*bHighlightFormFields .....	17
*FormFieldsHighlightAlpha .....	17
*FormFieldsHighlightColor.....	17
ActiveXVersion .....	17
<b>Methods.....</b>	<b>18</b>
1) Open and close PDF File.....	18
2) Viewing .....	20
3) Navigation .....	23
4) Search.....	28
5) Outline.....	30
6) Save.....	31
7) *Annotation .....	31
8) Form .....	32
9) Drawing.....	34
10) *Running Javascript .....	35
11) *HyperLink .....	35
12) Others .....	36

<b>Events</b> .....	<b>45</b>
BeforeDraw .....	45
AfterDraw .....	45
OnZoomChange .....	45
OnPageChange .....	45
OnOpenPassword .....	45
*OnHyperLink .....	46
OnSearchProgress .....	46
OnOpenFile .....	46
OnFilePathInvalidate .....	47
OnShowSavePrompt .....	47
OnOpenDocument .....	47
OnCloseDocument .....	47
OnDocumentChange .....	47
CustomFileGetSize .....	47
CustomFileGetBlock .....	48
*OnContextMenuIndex .....	48
OnClick .....	48
OnDbClick .....	48
OnRButtonClick .....	49
OnDownloadFinish .....	49
OnErrorOccurred .....	49
#FormFieldError .....	49
OnUploadFinish .....	49
OnTextHyperLink .....	50
*OnAddMenuItemAction .....	50
OnExcuteMenuItem .....	50
OnDoGoToRAction .....	50
<b>#PDFForm</b> .....	<b>52</b>
<b>Method</b> .....	<b>52</b>
AddField .....	52
RemoveFieldsByName .....	52
ExportToFDF .....	53
ImportFromFDF .....	53
GetFieldByIndex .....	53
GetFieldsCount .....	53
RemoveField .....	54
<b>#PDFFormField</b> .....	<b>55</b>
<b>Properties</b> .....	<b>55</b>
Alignment .....	55
BorderStyle .....	55
BorderWidth .....	55
ButtonLayout .....	55
CalcOrderIndex .....	56
CharLimit .....	56

DefaultValue .....	56
IsEditable.....	56
Behavior .....	56
IsHidden .....	57
IsMultiline .....	57
IsPassword.....	57
IsReadOnly.....	57
IsRequired.....	57
Name.....	58
NoViewFlag.....	58
PrintFlag.....	58
Style.....	58
TextFont.....	58
TextSize.....	59
Type.....	59
Value .....	59
Tooltip.....	60
Orientation.....	60
<b>Method.....</b>	<b>60</b>
PopulateListOrComboBox .....	60
SetBackgroundColor .....	60
SetBorderColor.....	61
SetForegroundColor .....	61
SetButtonCaption .....	61
SetButtonIcon.....	62
SetExportValues .....	62
SetJavaScriptAction .....	62
SetResetFormAction .....	63
SetSubmitFormAction.....	63
GetPageIndex .....	64
GetRectTop .....	64
GetRectLeft .....	64
GetRectRight.....	64
GetRectBottom.....	65
<b>^ CPDFPageAnnots .....</b>	<b>65</b>
<b>Method.....</b>	<b>65</b>
GetPageAnnots .....	65
GetAnnot .....	65
AddAnnot.....	65
RemoveAnnot.....	66
GetAnnotIndex .....	66
GetAnnotsCount.....	66
<b>^CPDFAnnot.....</b>	<b>67</b>
<b>Properties.....</b>	<b>67</b>
Thickness .....	67

BorderStyle .....	67
Color .....	67
LineStartingStyle .....	68
LineEndingStyle .....	68
FillColor .....	69
Opacity .....	69
Author .....	69
Subject .....	69
CreationDate .....	69
ModificationDate .....	70
Locked .....	70
Print .....	70
ReadOnly .....	70
Description .....	70
<b>Methods .....</b>	<b>71</b>
GetType .....	71
GetSubType .....	71
GetContents .....	71
SetContents .....	71
IsPopupOpen .....	71
SetPopupOpen .....	72
HasPopup .....	72
GetRect .....	72
SetRect .....	72
SetLinkGoToAction .....	73
SetLinkURLAction .....	73
DoAction .....	73
HasAction .....	73
GetMarkedState .....	74
SetMarkedState .....	74
GetReviewState .....	74
SetReviewState .....	74
GetMigrationState .....	75
SetMigrationState .....	75
SetStartingPoint .....	75
GetStartingPoint .....	75
SetEndingPoint .....	76
GetEndingPoint .....	76
SetMediaPoster .....	76
SetMultimedia .....	76
SetLinkQuadPoints .....	77
SetPolygonVertices .....	77
SetPencilVertices .....	77
AttachFile .....	77
<b>^CPDFFormatTool .....</b>	<b>78</b>

<b>Method</b> .....	<b>78</b>
GetFormatTool.....	78
SetFontName .....	78
GetFontName.....	78
SetFontSize.....	79
GetFontSize .....	79
SetFontColor .....	79
GetFontColor.....	79
SetBorderColor.....	80
GetBorderColor .....	80
SetFillColor.....	80
GetFillColor .....	80
SetFontBold.....	80
GetFontBold .....	81
GetFontBoldEnable .....	81
SetFontItalic.....	81
GetFontItalic .....	81
GetFontItalicEnable .....	82
SetAlign .....	82
GetAlign.....	82
SetCharSpace .....	82
GetCharSpace.....	83
SetCharHorzScale.....	83
GetCharHorzScale .....	83
<b>PDFPrinter</b> .....	<b>84</b>
<b>Properties</b> .....	<b>84</b>
PrinterName .....	84
printerRangeMode.....	84
printerRangeFrom .....	84
printerRangeTo.....	85
numOfCopies .....	85
Scaling.....	85
AutoRotate.....	85
AutoCenter .....	85
Collate .....	85
Rotation .....	86
RangeSubset.....	86
ReversePage.....	86
PageBorder .....	86
<b>Methods</b> .....	<b>86</b>
PrintWithDialog.....	86
PrintQuiet .....	86
SetPaperSize .....	87
<b>PDFOutline</b> .....	<b>88</b>
<b>Methods</b> .....	<b>88</b>

---

NavigateOutline .....	88
GetOutlineTitle.....	88
GetOutLineTitle2 .....	88
<b>PDFDocumentInfo.....</b>	<b>89</b>
<b>Properties .....</b>	<b>89</b>
Author.....	89
Subject .....	89
CreatedDate .....	89
ModifiedDate .....	89
Keywords.....	89
Creator .....	89
Producer.....	90
Title.....	90
<b>FindResult .....</b>	<b>91</b>
<b>Methods .....</b>	<b>91</b>
GetFindPageNum.....	91
GetFindFileName .....	91
GetFindString .....	91



## Overview

**Foxit PDF SDK ActiveX** is a visual programming component that offers PDF displaying capability with minimal resource demand and redistribution size. It can be easily integrated into a wide range of applications.

Foxit PDF SDK ActiveX uses the same parsing and rendering engine as [Foxit Reader](#). Therefore it can display PDF files with the same high quality and fast speed as Foxit Reader.

Compared to the DLL version of Foxit SDK, the ActiveX version is much easier to use and has much more rich features built inside. A programmer can simply drag and drop the component into their application and instantly add PDF displaying functionality. In addition, the ActiveX allows users to navigate, zoom, rotate, scroll and print out PDF documents.

Version 4.0 incorporates Annotation Module and many advanced PDF features. It enables you to build your own annotations and allows users to fill out, import or export PDF form data. Version 4.0 also introduces more functions and events, giving programmers flexible control over the component and more access to the PDF documents.

Foxit offers two versions of ActiveX 4.0 (standard and professional), which are provided with different GUID numbers that allows you to register both versions onto the same computer and access them with their own GUID. Compared with the professional version, the standard version doesn't include the following features: creating/editing annotation, importing/exporting form data, running JavaScript, converting PDF to text, etc. Based on your requirement needs you are able to pick and choose which ActiveX 4.0 version would be best for your application. You can purchase either the standard or professional version online. For the Form and Annotation Module, please contact [sales@foxitsoftware.com](mailto:sales@foxitsoftware.com) for licensing details. In this developer's guide, all the properties and functions marked with an asterisk (\*) are available only in the professional version, all the properties and functions marked with an asterisk (#) are available only in the Form Module and all the properties and functions marked with an asterisk (^) are available only in the Annotation Module.

Foxit PDF SDK ActiveX runs on Windows 95/NT or later. This is a standalone component and does not require any extra PDF software installed. Please note a user might need to have administrator rights to register the ActiveX under Windows successfully.

There are several complete demo programs written in different languages, including Visual Basic, Visual C++, Delphi, showing how to use the properties and methods of the ActiveX. To download these methods please visit [www.foxitsoftware.com](http://www.foxitsoftware.com).

Based on the professional version, Foxit PDF SDK ActiveX 4.0 adds many new interfaces to support PDF Form Fields, which will allow users to add, delete, import or export, form data in their PDF documents. The Form Module is licensed separately from ActiveX 4.0 Professional. If you would like to use this module, please contact us at

[sales@foxitsoftware.com](mailto:sales@foxitsoftware.com).

---

**UNLOCK Code:** If you have purchased Foxit PDF SDK ActiveX and received the full version of the ActiveX and the unlock code, you should call [UnLockActiveX](#) or [UnLockActiveXEx](#) functions once inside your program before you call ANY other functions of the ActiveX. This function is described in the Reference section. You don't need to call this function if you just want to evaluate the ActiveX.

GUID for Standard version: 0F6C092B-6E4C-4976-B386-27A9FD9E96A1

GUID for Professional version: F53B7748-643C-4A78-8DBC-01A4855D1A10

## Tutorials

The Foxit SDK ActiveX control comes with a single OCX file. To install it, please use command "regsvr32 OCX name". You may need to specify the proper path if OCX is not stored in current directory.

The ActiveX control handles user interface for you. It also supplies multiple properties and methods so that your application is able to control the ActiveX.

The following is a example, suppose we had already created an ActiveX control called FoxitReaderSDK.

### 1) Open a PDF File

We will open a PDF document named "testdoc.pdf"

// NOTE: If you are evaluating ActiveX, you don't need to unlock it,

// then evaluation marks will be shown on all PDF pages.

// For paid customers, please unlock the ActiveX first.

```
FoxitReaderSDK.UnLockActiveX("license_id","unlock_code");
```

```
FoxitReaderSDK.OpenFile("testdoc.pdf","");
```

### 2) Go to a specific page

We will go to the third page of the document

```
FoxitReaderSDK.GoToPage(2). //The index of first page is 0.
```

### 3) Zoom a page

If you want to show a PDF page with its original size, you can use the following code in VC:

```
FoxitReaderSDK.SetZoomLevel(0);
```

Or

```
FoxitReaderSDK. SetZoomLevel(100);
```

If you want to set the zoom factor to 200%, use the following code:

```
FoxitReaderSDK. SetZoomLevel(200);
```

For more information, see Function Reference section of this guide.

### 4) Rotate a page

A PDF page can be displayed in four directions: upright, rotated 90 degree, rotated 180 degree, or rotated 270 degree. To display it in different direction, you only need to call "SetRotate" () to change the Rotate property.

If you want to rotate the page (90 degrees) clockwise, you can use the following code in VC:

```
FoxitReaderSDK. SetRotate(1);
```

## 5) Print a PDF document

What you need to do is call “PrintWithDialog” method, a print dialog will pop up and you will be able to specify parameters and then print out the PDF document. If you want to print without popping up a dialog, you need to use the PDFPrinter interface.

## 6) Hide or show UI elements

You can call “ShowToolBar” to show or hide the toolbar. Likewise, you may call “ShowBookmark” to show or hide the bookmark panel and call “ShowStatusBar” to show or hide the status bar. If you prefer to build your own toolbar, you may hide the built-in ActiveX toolbar and then create your own outside the ActiveX.

## 7) Iterate the whole outline tree

You can call “GetOutlineFirstChild” or “GetOutlineNextSibling” to iterate the whole outline tree, allowing you to view the outline information from PDFOutline Interface.

## 8) Search a PDF document

You can call “FindFirst” to find the first instance of the given text in the whole document. If no occurrence is found, then the function returns 0. If an occurrence is found, the function returns a nonzero value and the occurrence will be highlighted. Select “FindNext” again to search for the next occurrence.

If you want to search for text inside PDF files without opening and displaying them, you may use the “FindFileFirst” or “FindFileNext”.

## 9) \*Annotations tools

End users can draw lines, circles and other shapes on a PDF document by using different markup tools. Your application may also change “CurrentTool” property programmatically, for example, to the “Line Tool”.

## 10) #Form Application

In version 3.0, Foxit ActiveX provides an interface for PDF forms. By calling “GetCurrentForm”, you can obtain the IForm interface pointer for the current PDF document. Then you can use properties and methods in IPDFForm and IPDFFormField interface classes to perform actions to PDF forms. The following code shows how to add a form button in a PDF page:

```
CPDFFormField button1 =  
form1.AddField("button1","button",m_nCurPage,0,0,55,30);  
button1.SetButtonCaption("N","Normal");  
button1.SetButtonCaption("R","Rollover");  
button1.SetButtonCaption("D","Down");  
button1.SetBehavior("push");// push;Invert; None; Outline;  
button1.SetTooltip("reset all form");  
button1.SetTextFont("Courier");
```

---

```
button1.SetTextSize(15);  
button1.SetJavaScriptAction("down","app.alert(\"Mouse Down!\")");
```

## 11) ^Annotation Application

In version 4.0, Foxit ActiveX provides some interfaces to operate Annotation objects. You can build your own annotations with these interfaces.

## Foxit PDF SDK ActiveX control

This section describes all properties and methods exposed by ActiveX. Please note that the reference shows everything in C syntax. If you use a programming language other than C/C++, you have to follow the syntax of that language.

Note: The functions marked with the (\*) only apply to the professional version, the functions marked with an asterisk (#) are available only in the Form Module and the functions marked with an asterisk (^) are available only in the Annotation Module.

### Properties

#### FilePath

Type:

BSTR, read-only

Description:

Full path to the current opened PDF file. If no PDF file is opened or the file is opened from the buffer or stream, the property is an empty string.

#### Password

Type:

BSTR, read-only

Description:

Password for the PDF.

#### PageCount

Type:

long, read-only

Description:

Total number of pages in the current open PDF file.

#### CurPage

Type:

long, read-only

Description:

Index of the current page of the PDF file. Page index starts from zero for the first page.

#### Rotate

Type:

short, read and write

Description:

Current rotate orientation, the value can be one of the following:

- 0 (normal);
- 1 (rotated 90 degrees clockwise);
- 2 (rotated 180 degrees);
- 3 (rotated 90 degrees counter-clockwise).

## Zoomlevel

Type:

long, read and write

Description:

Normally, the value of this zoom factor is between 10 and 1600. You may also use the following special values:

0=displaying the page in actual page size, this is the same as setting zoom level to 100%.

1=displaying the page with proper zoom level so that the whole page can be fit into the client window.

2=displaying the document with proper zoom level so that the width of the page fit to the client window.

## CurrentTool

Type:

BSTR, read and write

Description:

Read and set the current tool. The value can be set to one of following strings:

- "Hand Tool"
- "ZoomOut Tool"
- "ZoomIn Tool"
- "Select Text Tool"
- "Find Text Tool"
- "Snapshot Tool"
- \*"Loupe Tool"
- \*"Magnifier"
- \*"Annot Tool"
- \*"Rectangle Link Tool"
- \*"Quadrilateral Link Tool"
- \*"Arrow Tool"
- \*"Line Tool"
- \*"Dimension Tool"
- \*"Square Tool"
- \*"Rectangle Tool"
- \*"Circle Tool"
- \*"Ellipse Tool"
- \*"Polygon Tool"
- \*"Cloudy Tool"

- \*"Polyline Tool"
- \*"Pencil Tool"
- \*"Rubber Tool"
- \*"Highlight Tool"
- \*"Underline Tool"
- \*"Strikeout Tool"
- \*"Squiggly Tool"
- \*"Replace Tool"
- \*"Note Tool"
- \*"Push Button Tool"
- \*"Check Box Tool"
- \*"Radio Button Tool"
- \*"Combo Box Tool"
- \*"List Box Tool"
- \*"Text Field Tool"
- \*"Distance Tool"
- \*"Perimeter Tool"
- \*"Area Tool"
- \*"Typewriter"
- \*"CallOut"
- \*"Textbox"
- \*"Image Tool"
- \*"Sound Tool"
- \*"Movie Tool"
- \*"FileAttachment Tool"
- \*"Attach a file"

And so on.

You can call `CountTools` to learn how many tools are available in the current version of ActiveX, and then call `GetToolByIndex` to get the tool names.

## Printer

Type:

IPDFPrinter, read-only

Description:

Printer property returns an [IPDFPrinter](#) interface that you can use for managing the printer and sending the printout.

## DocumentInfo

Type:

IPDFDocumentInfo\*, read-only

Description:

DocumentInfo property returns an [IPDFDocumentInfo](#) interface with which you can use to retrieve document information, such as Author, Creator, Creation Date, Keywords, ModDate,



Producer Subject and Title.

### **\*bHasFormFields**

Type:

BOOL , read-only

Description:

If current document contains form fields, then bHasFormFields is True; otherwise, it is False.

### **\*bHighlightFormFields**

Type:

BOOL , read and write

Description:

Setting bHighlightFormFields to True will highlight all interactive form fields thereby having a better visual effects.

### **\*FormFieldsHighlightAlpha**

Type:

short, read and write

Description:

Represent 256 levels of transparency of form field highlight color.  
0=transparent; 255=opaque.

### **\*FormFieldsHighlightColor**

Type:

OLE\_COLOR, read and write

Description:

Represent highlight color of form fields.

### **ActiveXVersion**

Type:

BSTR, read-only

Description:

Get the Version info of ActiveX control.

## Methods

### 1) Open and close PDF File

#### OpenFile

Open a PDF file from a local disk or from an http server.

Prototype:

```
BOOL      OpenFile (BSTR FilePath, BSTR Password)
```

Parameters:

- FilePath - Path to the PDF file or URL to a HTTP server.
- Password - Password for the PDF. If no password, specify an empty string.

Return value:

Non-zero if the PDF file is successfully opened, otherwise it is zero.

Comment:

The file will not be locked if it is opened by this method. It can be opened by other program.

#### OpenMemFile

Open a PDF file that is stored in memory.

Prototype:

```
BOOL      OpenMemFile(long pBuffer, long Size, BSTR Password)
```

Parameters:

- pBuffer - Caller-supplied pointer to a buffer containing PDF data .
- Size - Size of the buffer pointed to by pBuffer.
- Password - Password for the PDF. If no password, specify an empty string.

Return value:

Nonzero if the PDF file is successfully opened, otherwise it is zero.

#### OpenBuffer

Open a PDF file from the buffer.

Prototype:

```
BOOL      OpenBuffer(VARIANT Buffer, long size, BSTR password);
```

Parameters:

- Buffer - Byte array containing the PDF.
- Size - Size of the byte array.
- Password - Password to open the PDF document

Return value:

Nonzero if the PDF file is successfully opened, otherwise it is zero.

#### OpenStream

Open a PDF file from the IStream interface..

Prototype:

```
BOOL      OpenStream (IStream* Stream, BSTR Password)
```

Parameters:

- Stream - An IStream interface.
- Password - Password for the PDF. If no password, specify an empty string.

Return value:

Nonzero if the PDF file is successfully opened, otherwise it is zero.

### OpenCustomFile

Open a PDF document from a custom access descriptor. When your program calls this method, ActiveX will trigger the CustomFileGetSize and CustomFileGetBlock events. Inside the event handler, your program will open the PDF document from a custom format; return the file size and block of data. See the description of CustomFileGetSize and CustomFileGetBlock for more details.

Prototype:

```
BOOL      OpenCustomFile(BSTR Password)
```

Parameter:

Password - Password for the PDF. If no password, specify an empty string.

Return value:

Non-zero if the PDF file is successfully opened, otherwise it is zero.

### OpenFtpFile

Open a PDF file from an FTP server.

Prototype:

```
BOOL      OpenFtpFile(BSTR ftpName, BSTR username, BSTR userPassword,  
                      long port, BSTR filePath, BSTR filePassword, boolean Passive);
```

Parameter:

ftpName - FTP server name  
userName - FTP Username  
userPassword - FTP password  
port - Port on FTP server  
filePath - FTP file Path  
filePassword - PDF file password  
Passive - Passive or active connection

Return value:

One (1) if the PDF file is successfully opened, otherwise it is zero (0).

### \*UploadCurFileToFTP

Upload the current PDF file to an FTP server

Prototype:

```
BOOL      UploadCurFileToFTP(BSTR ftpName, BSTR userName, BSTR userPassword,  
                              long port, BSTR FilePath);
```

Parameter:

ftpName - FTP server name  
userName - FTP Username  
userPassword - FTP password  
port - Port on FTP server  
filePath - FTP file Path

Return value:

One (1) if the PDF file is successfully opened, otherwise it is zero (0).

### **CloseFile**

Close the currently loaded PDF file.

Prototype:

```
Void CloseFile()
```

Parameter:

[None]

Return value:

[None]

### **SetFileStreamOption**

Set the file stream option when opening the file.

Prototype:

```
Void SetFileStreamOption(BOOL bFileStream);
```

Parameter:

bFileStream - A BOOL value indicating the file stream option.

Return value:

[None]

Comment:

Loading the stream contents into memory will improve performance if the file is frequently accessed. However it will cause more memory consumption.

## **2) Viewing**

### **ShowTitleBar**

Show or hide the title bar;

Prototype:

```
void ShowTitleBar(BOOL bShow);
```

Parameters:

bShow - If this parameter is FALSE, the title bar will be invisible.  
If this parameter is TRUE, the title bar will be visible.

Return value:

[None]

### **ShowToolBar**

Show or hide the toolbar;

Prototype:

```
void ShowToolBar(BOOL bShow);
```

Parameters:

bShow - If this parameter is FALSE, the toolbar will be invisible.  
If this parameter is TRUE, the toolbar will be visible.

Return value:

[None]

### ShowToolBarButton

Show or hide the toolbar button

Prototype:

```
void ShowToolBarButton (short nIndex, BOOL bShow)
```

Parameters:

- nIndex - The index of the button.
- bShow - If this parameter is FALSE, the toolbar button will be invisible.  
If this parameter is TRUE, the toolbar button will be visible.

Return value:

[None]

### ShowStatusBar

Show or hide the status bar

Prototype:

```
void ShowStatusBar(BOOL bShow);
```

Parameters:

- bShow - If this parameter is FALSE, the Status Bar will be invisible.  
If this parameter is TRUE, the Status Bar will be visible.

Return value:

[None]

### ShowNavPanelByString

Show a navigation panel by its name.

Prototype:

```
BOOL ShowNavPanelByString(LPCTSTR lpszPanelName)
```

Parameters:

- lpszPanelName - The panel name, including Bookmark panel, Pages panel, Layer panel, and Attachments panel.

Return Value:

Returns True if successful, False otherwise.

### \*ShowFormFieldsMessageBar

Show or hide the FormFieldsMessageBar.

Prototype:

```
void ShowFormFieldsMessageBar(BOOL bShow)
```

Parameters:

- bShow - If this parameter is FALSE, the FormFieldsMessageBar will be invisible.  
If this parameter is TRUE, the FormFieldsMessageBar will be visible.

Return value:

[None]

### SetLayoutShowMode

Set the page layout. A PDF document can be displayed as n columns by m rows. No matter what the facing count number is, when the page layout is set to MODE\_SINGLE, the

ActiveX window will display one row at a time, when the page layout is set to `MODE_CONTINUOUS`, the window will be able to display adjacent rows at the same time.

Prototype:

```
Void SetLayoutShowMode (BrowseMode nShowMode, short nFacingCount);
```

Parameters:

`nShowMode` - the value can be set as following:  
`MODE_SINGLE = 0.`  
`MODE_CONTINUOUS = 1.`

`nFacingCount` - Number of columns.

Return value:

[None]

### **GetLayoutShowMode**

Obtain the current layout mode.

Prototype:

```
Void GetLayoutShowMode(short* pnShowMode, short* pnFacingCount)
```

Parameters:

`pnShowMode` - Toggles to the current display mode.  
`pnFacingCount` - Toggles to the number of pages that are displayed horizontally.

Return value:

[None]

### **SetFacingCoverLeft**

When in browser facing mode, this function will set cover page of the PDF document to be rendered on the left.

Prototype:

```
Void SetFacingCoverLeft (BOOL bLeft)
```

Parameters:

`bLeft` - A BOOL value indicating whether to set cover page to be rendered on the left.

Return value:

[None]

### **ShowNavigationPanels**

Toggle between the navigation panel

Prototype:

```
BOOL ShowNavigationPanels(BOOL bShow)
```

Parameters:

`bShow` - TRUE shows the navigation panel, FALSE hides it.

Return value:

Returns True if successful, False otherwise.

### **GetPanelStatus**

Obtain the status of navigation panels.

Prototype:

```
BOOL GetPanelStatus()
```

Parameters:

[None]

Return value:

Returns TRUE if the panel is visible, FALSE for else.

### **EnableToolTip**

Toggle between the displaying of Tool Tips.

Prototype:

```
BOOL EnableToolTip (BOOL bEnable)
```

Parameters:

bEnable - TRUE shows the Tool Tips, FALSE hides them

Return value:

Returns True if successful, False otherwise.

## **3) Navigation**

### **ExistForwardStack**

Detect the existence of next view.

Prototype:

```
BOOL ExistForwardStack ();
```

Parameters :

[None]

Return value:

If next view exists, then it will return true. Otherwise, it will return false.

Comment:

Quite often, when a user navigates within a PDF file, he would like to go back to a previous reading point. View is a concept that defines certain reading point or displaying status. Certain user actions will create new views. For example, if a user turns to a new page, and then zoom in the page, these two actions will create two new views. A program may call this group of methods to allow user to jump among different views conveniently.

### **GoForwardStack**

Jump to the next view.

Prototype:

```
Void GoForwardStack ();
```

Parameters:

[None]

Return value:

[None]

### **ExistBackwardStack**

Detect the existence of previous view.

Prototype:

BOOL ExistBackwardStack ();

Parameters:

[None]

Return value:

If previous view exists, then it will return true. Otherwise, it will return false.

### GoBackwardStack

Jump to previous view,

Prototype:

void GoBackwardStack ();

Parameters:

[None]

Return value:

[None]

### SetViewRect

Display a rectangle of current PDF page.

Prototype:

VoidSetViewRect (float Left, float Top, float Width, float Height);

Parameters:

- Left - The horizontal coordinate of the top left corner.
- Top - The vertical coordinate of the top left corner.
- Width - The width of the rectangle.
- Height - The height of the rectangle.

Return value:

[None]

Comment:

This function will show a rectangle of current PDF page. The coordinate here is PDF coordinate, not device coordinate. And the unit is PDF point. The function will keep the position and size of ActiveX window unchanged and adjust the position and the zoom factor of current PDF page so that the designate rectangle of current PDF page will be shown fully inside the ActiveX window. A typical application is: the end user use the mouse to click and drag a rectangle and then release the mouse, the program will call ConvertClientCoodToPageCood to convert the mouse coordinates into PDF coordinates and then call SetViewRect to display the specified area in full view.

### ConvertClientCoordToPageCoord

Converts a point in ActiveX control window's client co-ordinates into PDF page coordinate.

Prototype:

BOOL ConvertClientCoordToPageCoord (long nClientX, long nClientY,  
long\* pnPageIndex, float\* pPageX, float\* pPageY);

Parameters:

- nClientX - X coordinate in the ActiveX control window's client co-ordinates, in pixels



- nClientY - Y coordinate in the ActiveX control window's client co-ordinates, in pixels
- pnPageIndex - For returning page number in which the given point falls on
- pPageX - For returning x coordinate of the point inside the PDF page (in PDF co-ordinate system)
- pPageY - For returning y coordinate of the point inside the PDF page (in PDF co-ordinate system)

**Return value:**

Return value indicates whether the conversion is successful. The client area contains the PDF page being shown as well as some grey background. If the point is located in the grey background, the conversion will fail.

### **ConvertClientCoordToPageCoordEx**

The extension of ConvertClientCoordToPageCoord function.

**Prototype:**

```

BOOL ConvertClientCoordToPageCoordEx (long nClientX, long nClientY,
                                       VARIANT* pnPageIndex, VARIANT* pPageX,
                                       VARIANT* pPageY);

```

**Parameters:**

- nClientX - X coordinate in the ActiveX control window's client co-ordinates, in pixels
- nClientY - Y coordinate in the ActiveX control window's client co-ordinates, in pixels
- pnPageIndex - For returning page number on which the given point falls
- pPageX - For returning x coordinate of the point inside the PDF page (in PDF co-ordinate system)
- pPageY - For returning y coordinate of the point inside the PDF page (in PDF co-ordinate system)

**Return value:**

Return value indicates whether the conversion is successful or not. The client area contains the PDF page being shown as well as some grey background. If the point is located in the grey background, the conversion will fail.

### **ConvertPageCoordToClientCoord**

Converts PDF page coordinates to the coordinates inside ActiveX control window's client area.

**Prototype:**

```

BOOL ConvertPageCoordToClientCoord (long nPageIndex, float dPageX, float dPageY,
                                     long* pnClientX, long* pnClientY);

```

**Parameters:**

- nPageIndex - page number
- dPageX - X coordinate inside the PDF page (in PDF co-ordinate system)
- dPageY - Y coordinate inside the PDF page (in PDF co-ordinate system)
- pnClientX - For returning X coordinate in the ActiveX control window's client area. A

negative result indicates that the point is outside the ActiveX control window's client area.

- pnClientY - For returning Y coordinate in the ActiveX control window's client area. A negative result indicates that the point is outside the ActiveX control window's client area.

Return value:

Return value indicates whether the conversion is successful. If the document is not opened properly or if the page number is incorrect, then the return value will be false. Otherwise, the return value will be true.

### **GotoPageDest**

Go to a specified position in a PDF document.

Prototype:

```
Void GotoPageDest (ILink_Dest * link_dest);
```

Parameters:

- link\_dest - An ILink\_Dest interface you get from event OnHyperLink .

Return value:

[None]

### **GoToPagePos**

Go to a specified position in a PDF document.

Prototype:

```
Void GoToPagePos (long nPageIndex, float PageX, float PageY);
```

Parameters:

- nPageIndex - Index of the page you want to view.  
 PageX - Specify the x coordinate of the position inside the PDF page which index is specified by nPageIndex.(in PDF co-ordinate system)  
 PageY - Specify the y coordinate of the position inside the PDF page which index is specified by nPageIndex.

Return value:

[None]

### **GetVisibleLeftTopPage**

Get the page number of the view at the top left side

Prototype:

```
Long GetVisibleLeftTopPage ();
```

Parameter:

[None]

Return value:

the page index which showed on the left top.

### **ScrollView**

Scroll the current view by dx, dy, the unit is device pixel.

Prototype:

Void ScrollView (long dx, long dy);

Parameters:

- dx - The horizontal distance of the scrolling action.
- dy - The vertical distance of the scrolling action.

Return value:

[None]

### **GetScrollLocation**

Get the current scroll location in current page.

rototype:

void GetScrollLocation (long \*dx, long \*dy);

Parameters:

- dx - For returning x coordinate of the current scroll location
- dy - For returning y coordinate of the current scroll location

Return value:

[None]

### **GetScrollLocationEx**

The extension of GetScrollLocation function.

rototype:

void GetScrollLocationEx (VARIANT \* HPos, VARIANT \* VPos);

Parameters:

- HPos - For returning x coordinate of the current location you scroll to
- VPos - For returning y coordinate of the current location you scroll to

Return value:

[None]

### **GoToPage**

Goes to the specific page of the currently opened PDF document

Prototype:

void GoToPage( long page\_index )

Parameters:

- page\_index - The specific page number

Return value:

[None]

### **GoToNextPage**

Traverse to the next page of the currently open PDF document.

Prototype:

VoidGoToNextPage ();

Parameters:

[None]

Return value:

[None]

Note:

This function will be removed in further version and you can use function GoToPage instead.

### GoToPrevPage

Traverse to the previous page of the currently open PDF document.

Prototype:

```
Void GoToPrevPage ();
```

Parameters:

[None]

Return value:

[None]

Note:

This function will be removed in further version and you can use function GoToPage Instead

## 4) Search

### FindFirst

Search the document for a string. If the function finds the first occurrence of the string, it will jump to the page, update CurPage property, highlight the occurrence and return true value. Otherwise it will return false.

Prototype:

```
BOOL FindFirst (BSTR search_string, BOOL bMatchCase, BOOL bMatchWholeWord)
```

Parameters:

- SearchString - The string you want to search.
- BMatchCase - Case sensitive or not.
- BMatchWholeWord - Search for whole word only or not.

Return value:

Nonzero if an occurrence of the string is found, otherwise it will be zero.

### FindFirstEx

The extension of the FindFirst function. Provides an interface to search for a string in the document.

Prototype:

```
BOOL FindFirstEx(const VARIANT FAR& search_string, BOOL bMatchCase,  
                BOOL bMatchWholeWord)
```

Parameters:

- Search\_string - The string you want to search.
- bMatchCase - Case sensitive or not.
- bMatchWholeWord - Search for whole word only or not.

Return Value:

Nonzero if an occurrence of the string is found, otherwise it will be zero.

### FindNext

Search for the next occurrence of the given string in the whole document. If the ActiveX find the next occurrence, it will jump to the page, update CurPage property, highlight the occurrence and return true value. Otherwise, it will return false. Please note that FindNext will use the same searching criteria specified in the FindFirst method, including bMatchCase, bMatchWholeWord. If bSearchDown is true, then the search goes down the document. If bSearchDown is false, then the search goes up.

Prototype:

```
BOOL FindNext (BOOL bSearchDown);
```

Parameters:

bSearchDown - Search down (True) or up (False).

Return value:

If next occurrence is found, the return value will be non-zero, otherwise it will be zero.

### FindFileFirst

Search a file for a string and returns an IFindResult interface if it finds the string. Otherwise it will return null. This method allows you to search a file without first opening it. For example, if you want to search for a keyword in all the PDF files inside a folder, you may iterate all the PDF files inside that folder and search them one by one for the keyword. If the ActiveX find an occurrence inside a PDF file, it will return IFindResult which contains all the details of the occurrence. Then you can use GoToSearchResult to open the file, jump to the page and highlight the occurrence.

Prototype:

```
IFindResult* FindFileFirst(BSTR file_path, BSTR search_string, BOOL bMatchCase,
                           BOOL bMatchWholeWord)
```

Parameters:

file\_path - Path to the PDF file.  
 search\_string - The string you want to search.  
 bmatchCase - Case sensitive or not.  
 BMatchWholeWord - Search for whole word only or not.

Return value:

An IFindResult interface if an occurrence is found. Otherwise it will be null.

### FindFileFirstEx

The extension of FindFileFirst function.

Prototype:

```
IFindResult * FindFileFirstEx(BSTR file_path, BSTR password, VARIANT search_string,
                              boolean bMatchCase, boolean bMatchWholeWord);
```

Parameters:

file\_path - Path to the PDF file.  
 Password - The password for the searched PDF file.  
 search\_string - The string you want to search.  
 bMatchCase - Case sensitive or not.  
 BMatchWholeWord - Search for whole word only or not.

Return value:

An IFindResult interface if an occurrence is found. Otherwise it will be null.

### FindFileNext

Search for the next occurrence of the given string in the file specified by FindFileFirst.

Prototype:

```
IFindResult * FindFileNext ();
```

Parameters:

[None]

Return value:

If the next occurrence is found, the return value will be IFindResult. Otherwise it will be null.

### GoToSearchResult

Display and highlight the search result.

Prototype:

```
Void GoToSearchResult (IFindResult* findresult);
```

Parameters:

Findresult - An IFindResult interface returned by FindFileFirst, or FindFileNext .

Return value:

[None]

### \*SearchAndHighlightAllTextOnPage

Highlight all instance of a given keyword in a specified page

Prototype:

```
void SearchAndHighlightAllTextOnPage(BSTR searchstring, BOOL bMatchCase,
                                      BOOL bMatchWholeWord,long pageNo);
```

Parameters:

PageNo - Number of the page you want to search.

Return Value:

[None]

## 5) Outline

### GetOutlineFirstChild

Get first child item of the current outline.

Prototype:

```
IPDFOutline* GetOutlineFirstChild( IPDFOutline* Outline)
```

Parameters:

Outline - The parent item whose first child item will be returned.  
If you want to get the root item of the outline tree, set null as the parameter value.

Return value:

If the specified item has child items, then the first child item will be returned.  
Otherwise null will be returned.

### GetOutlineNextSibling

Get next sibling item.

Prototype:

IPDFOutline\* GetOutlineNextSibling ( IPDFOutline\* Outline)

Parameters:

Outline - The outline item whose next sibling will be returned

Return value:

If the next sibling item exists, it will be returned. Otherwise, null will be returned.

## 6) Save

### SaveAs

Save the currently loaded PDF document into a file.

Prototype:

Void SaveAs (BSTR FileName)

Parameters:

FileName - Specifies the name of the file to be saved.

Return value:

[None]

### Save

Save the currently loaded PDF document.

Prototype:

Void Save ()

Parameters:

[None]

Return value:

[None]

### SaveToStream

Save the currently loaded PDF document into memory.

Prototype:

IStream\* SaveToStream()

Parameters:

[None]

Return value:

An IStream interface supports reading and writing data to stream objects which contain the PDF file data.

## 7) \*Annotation

### \*ExportAnnotsToFDFFile

Export comments from current document to a Form Data Format (FDF) file.

Prototype:

BOOL ExportAnnotsToFDFFile(BSTR FDFFileName)

Parameters:

FDFFileName - The FDF file path.

Return Value:

Return value indicates whether the operation is successful.

### **\*ImportAnnotsFromFDFFile**

Import comments from a Form Data Format (FDF) file to current document

Prototype:

BOOL ImportAnnotsFromFDFFile(BSTR FDFFileName)

Parameters:

FDFFileName - The FDF file path.

Return Value:

Return value indicates whether the operation is successful.

### **\*SetBDrawAnnot**

Set the annotation flag.

Prototype:

void SetBDrawAnnot(BOOL bDrawAnnot);

Parameters:

bDrawAnnot - If this parameter is true, it can draw annotation.  
If this parameter is false, it can not draw annotation.

Return Value:

[None]

### **\*ShowAllPopup**

Popup all annotations.

Prototype:

Void ShowAllPopup (BOOL bShow);

Parameters:

bShow - If this parameter is true, the annotation will pop up.

Return Value:

[None]

## **8) Form**

### **\*ExportFormToFDFFile**

Export PDF form data to a Form Data Format (FDF) file.

Prototype:

BOOL ExportFormToFDFFile (BSTR FDFFileName)

Parameters:

FDFFileName - The FDF file path.

Return Value:

Return value indicates whether the operation is successful.

### **\*ImportFormFromFDFFile**

Import data from a Form Data Format (FDF) file into PDF forms.



Prototype:

```
BOOL ImportFormFromFDFFile(BSTR FDFFileName)
```

Parameters:

FDFFileName - The FDF file path.

Return Value:

Return value indicates whether the operation is successful.

### **\*FindFormFieldsTextFirst**

Search the text in form fields.

Prototype:

```
BOOL FindFormFieldsTextFirst (BSTR searchstring, BOOL bMatchCase);
```

Parameters:

Searchstring - The string you want to search.

bMatchCase - Case sensitive or not.

Return value:

Return value indicates whether the searched string is found.

### **\*FindFormFieldsTextNext**

Search the text in form fields.

Prototype:

```
BOOL FindFormFieldsTextNext ()
```

Parameters:

[None]

Return value:

Return value indicates whether the searched string is found.

### **\* SubmitForm**

Submit PDF form field to the specific destination.

Prototype:

```
BOOL SubmitForm(BSTR csDestination)
```

Parameters:

csDestination - The input string contains the submit URL

Return value:

Returns TRUE if successful, FALSE for failed.

### **#GetCurrentForm**

Obtain the pointer of the Form interface. This is needed to add new PDF Form elements.

Prototype:

```
IPDFForm* GetCurrentForm()
```

Parameters:

[None]

Return value:

Returns the Form interface point if successful, Null otherwise.

## 9) Drawing

### \*AddWaterMark

Insert a text as watermark into the document.

Prototype:

```
BOOL AddWaterMark (short page, BSTR string, float left, float bottom,
                  short fontsize, OLE_COLOR fontcolor,
                  short textmode, short alpha, short rotate);
```

Parameters:

- Page - The page number of the document to be Added watermark.
- String - The text that is displayed as watermark.
- Left - The horizontal X position in which the watermark is placed.
- Bottom - The horizontal y position in which the watermark is placed
- Fontsize - The text size.
- Fontcolor - The text color.
- Textmode - The value must be 0, 1, 2.
  - 0 means fill the text,
  - 1 means stroke the text,
  - 2 means fill then stroke the text.
- alpha - A number from 0 to 255, identifying the alpha value.
- rotate - The angle through which the watermarks are to be rotated.

Return value:

Return value indicates whether the watermark is added.

### \*AddImageObject

Insert an image into the document.

Prototype:

```
BOOL AddImageObject (long nPageIndex, float left, float bottom, float width,
                    float height, BSTR BmpFileName, short alpha, short rotate);
```

Parameters:

- nPageIndex - The page number of the document to be Added image.
- left - The horizontal X position in which the image is placed.
  - Starting from 0 at the left-most pixel.
- bottom - The horizontal y position in which the image is placed
  - Starting from 0 at the bottom-most scan line.
- width - The width of the bitmap.
- height - The height of the bitmap.
- BmpFileName - The file path of the image.
- alpha - A number from 0 to 255, identifying the alpha value.
- rotate - The angle through which the image are to be rotated.

Return Value:

Return value indicates whether the image is added.

## 10) \*Running Javascript

### \*ShowDocJsDialog

Popup Document Javascript Dialog.

Prototype:

```
void ShowDocJsDialog();
```

Parameters:

[None]

Return Value:

[None]

### \*ShowJsConsoleDialog

Popup Javascript Console Dialog.

Prototype:

```
void ShowJsConsoleDialog();
```

Parameters:

[None]

Return Value:

[None]

## 11) \*HyperLink

### \*CountHyperLinks

Compute the number of HyperLinks in a PDF page

Prototype:

```
short CountHyperLinks(short nPageIndex);
```

Parameters:

nPageIndex - the specific page number of the PDF file

Return Value:

Returns the link count if successful, 0 for no links, -1 for failure.

### \*HighlightHyperLink

Highlight a specific HyperLink in a PDF page

Prototype:

```
void HighlightHyperLink(short nPageIndex, short nLinkIndex)
```

Parameters:

nPageIndex - the specific page number

nLinkIndex - index of the HyperLink

Return Value:

[None]

### \*GetHyperLinkRect

Obtain the position of a specific HyperLink

Prototype:

```
BOOL GetHyperLinkRect(short nPageIndex, short nIndex, float* top, float* left,
```

float\* bottom, float\* right)

Parameters:

- nPageIndex - the specific page number
- nLinkIndex - index of the HyperLink
- top - returned pointer for top coordinate
- left - returned pointer for left coordinate
- bottom - returned pointer for bottom coordinate
- right - returned pointer for right coordinate

Return Value:

Returns True if successful, False otherwise.

### **\*GetHyperLinkInfo**

Obtain the link information of a specific HyperLink.

Prototype:

```
BOOL GetHyperLinkInfo(short nPageIndex, short nIndex, BSTR* linktype,
                      BSTR* linkdata, L PDISPATCH* linkdest)
```

Parameters:

- nPageIndex - the specific page number
- nLinkIndex - index of the HyperLink
- linktype - returned pointer containing the type of HyperLink
- linkdata - returned pointer containing the String data of the HyperLink
- linkdest - returned pointer containing the redirection destination of the HyperLink

Return value:

Returns True if successful, False otherwise.

Note:

Currently only the link type of GoTo, GoToR, Lanuch and URI can be obtained.

### **\*EnableHyperLink**

Enable/Disable HyperLinks.

Prototype:

```
Void EnableHyperLink(BOOL bEnable)
```

Parameters:

- bEnable - TRUE enables Hyperlinks, FALSE disables them.

Return value:

[None]

## **12) Others**

### **GetSelectedText**

Get currently selected text.

Prototype:

```
BSTR GetSelectedText ();
```

Parameters:

[None]

Return value:

The text that has been selected.

### **GetSelectedTextEx**

The extension of GetSelectedText function.

Prototype:

```
long GetSelectedTextEx(long* pBuffer, long nBuflen);
```

Parameters:

pBuffer - The input buffer to receive the selected text  
nBufLen - The input size of the buffer.

Return value:

Return the buffer length of the selected text.

### **OpenFileForPrinter**

Print a PDF file without displaying it.

Prototype:

```
IPDFPrinter* OpenFileForPrinter(BSTR file_path)
```

Parameters:

file\_path - Path to the PDF file (including file extension).

Return value:

Interface of IPDFPrinter that you can use to control the printer.

### **OpenMemFileForPrinter**

Print a memory PDF file without displaying it.

Prototype:

```
IPDFPrinter* OpenMemFileForPrinter(long buffer, long size)
```

Parameters:

buffer - The buffer of PDF file.  
size - The size of the buffer.

Return value:

Interface of IPDFPrinter that you can use to control the printer.

### **\*Highlight**

Highlight a specified rectangular region on the specified page of this document.

Prototype:

```
void Highlight(long nPageIndex, float left, float top, float right, float bottom)
```

Parameters:

nPageIndex - Number of the page where the specified rectangular region is to be highlighted  
left - X-coordinate of the bottom left corner of the rectangular region  
top - Y-coordinate of the bottom left corner of the rectangular region  
right - X-coordinate of the top right corner of the rectangular region  
bottom - Y-coordinate of the top right corner of the rectangular region

Return value:

[None]

**\*RemoveAllHighlight**

Remove all highlight in current open document.

Prototype:

```
void RemoveAllHighlight()
```

Parameters:

[None]

Return value:

[None]

**GetPageText**

Extract text content from a PDF page of the currently loaded PDF file.

Prototype:

```
BSTR GetPageText (long nPageIndex)
```

Parameters:

nPageIndex - Number of the page you want to extract text from.

Return value:

The text that has been extracted.

**\*GetPageTextW**

Extracts text content from a page in the current loaded PDF file.

Prototype:

```
long GetPageTextW(long nPageIndex, long FAR* pBuffer, long FAR* nBuflen)
```

Parameters:

nPageIndex - Number of page you want to extract text from  
pBuffer - The buffer to place the desired content of the page  
nBuflen - The length of the buffer

Return Value:

Returns -1 if successful, 0 otherwise.

**CountTools**

Get the number of tools that can be used in the current version of ActiveX.

Prototype:

```
short CountTools()
```

Parameters:

[None]

Return Value:

Number of tools.

**GetToolByIndex**

Get the name of a tool.

Prototype:

```
BSTR GetToolByIndex (short nIndex)
```

Parameters:

nIndex - The range of nIndex is:  $0 \leq nIndex < \text{CountTools}()$ .

Return Value:

The name of the tool is returned.

### **GetDocPermissions**

Get file permission flags of the document.

Prototype:

long GetDocPermissions ()

Parameter:

[None]

Return value:

A 32-bit integer indicates permission flags. Please refer to PDF Reference for detailed description, if the document is not protected, 0xffffffff will be returned.

### **ShowDocumentInfoDialog**

Popup Document Properties Dialog.

Prototype:

void ShowDocumentInfoDialog()

Parameter:

[None]

Return value:

[None]

### **SetModulePath**

Set the path of fpdfcjk.bin file

Prototype:

void SetModulePath(LPCTSTR lpFolderName)

Parameters:

lpFolderName - The path of fpdfcjk.bin file

Return Value:

[None]

### **SetCurrentLanguage**

The user interface of ActiveX can be switched to one of the 30+ languages dynamically.

This requires extra language file (in xml format) to be accompanied with the ActiveX. If you want a specific language file, please contact us at [sales@foxitsoftware.com](mailto:sales@foxitsoftware.com).

Prototype:

void SetCurrentLanguage(short LanguageID);

Parameters:

LanguageID - Language identifier.

A value from 0 to 30 to represent different languages.

Return value:

[None]

## SetCurrentLanguageByString

The user interface of ActiveX can be switched to one of the 30+ languages dynamically. This requires extra language file (in xml format) to be accompanied with the ActiveX. If you want a specific language file, please contact us at [sales@foxitsoftware.com](mailto:sales@foxitsoftware.com).

Prototype:

```
void SetCurrentLanguageByString(BSTR FileName);
```

Parameters:

FileName - The file path of the language file.

Return value:

[None]

## SetCurrentWnd

By the use of this function, users can set the current instance when ActiveX runs in Multi-instance.

Prototype:

```
void SetCurrentWnd(long hWnd);
```

Parameters:

hWnd - The HWND of a OCX instance.

Return value:

[None]

## SetLogFile

Users can call this function to set a log file in your application and each function you called will be recorded to this log file.

Prototype:

```
BOOL SetLogFile(BSTR filepath);
```

Parameters:

filepath - the log file's path.

Return value:

A bool value specifies whether the log file is set.

## SetCrashLog

Users can call this function to set a log file in your application. The crash issue you counter will be recorded to this log file.

Prototype:

```
BOOL SetCrashLog (BSTR filepath);
```

Parameters:

filepath - the log file's path.

Return value:

A bool value specifies whether the log file is set or not.

## \*IsDualPage

Check the type of the page.

Prototype:



BOOL IsDualPage(short pageIndex);

Parameters:

pageIndex - The page number of the document.

Return value:

A bool value specifies whether the page is two layers.

Two layers mean a page has an image with hidden text.

### **\*ExportPagesToPDF**

Export some pages in current document into a pdf file.

Prototype:

BOOL ExportPagesToPDF (BSTR lpszPDFFileName, BSTR lpszPageRangeString);

Parameters:

lpszPDFFileName - The file path to insert the page.

lpszPageRangeString - The page range string. Such as "0, 2, 3-5", please note that "5-2" is invalid. In other words, the value before the dash not allows being larger than the after one.

Return value:

A bool value specifies the pages have been exported.

### **\*GetBitmap**

Render contents in a page as a bitmap

Prototype:

long GetBitmap(short nPageIndex, long pixelWidth, long pixelHeight, float rectLeft, float rectTop, float rectRight, float rectBottom, long PixelFormat);

Parameters:

nPageIndex - The page number of the document.

pixelWidth - The width of the bitmap.

pixelHeight - The height of the bitmap.

rectLeft - Left pixel position of the display area in the device coordination.

rectTop - Top pixel position of the display area in the device coordination

rectRight - Right pixel position of the display area in the device coordination.

rectBottom - Bottom pixel position of the display area in the device coordination.

PixelFormat - The pixel format of the bitmap.

Return value:

The handle of the bitmap.

### **GetPageHeight**

Get page height

Prototype:

float GetPageHeight(short nPageIndex);

Parameters:

nPageIndex - The page number of the document.

Return value:

Page height (excluding non-displayable area) measured in points.

One point is 1/72 inch (around 0.3528 mm)

### **GetPageWidth**

Get page width

Prototype:

```
float GetPageWidth(short nPageIndex);
```

Parameters:

nPageIndex - The page number of the document.

Return value:

Page width (excluding non-displayable area) measured in points.

One point is 1/72 inch (around 0.3528 mm)

### **AboutBox**

Popup the about box.

Prototype:

```
void AboutBox();
```

Parameters:

[None]

Return value:

[None]

### **PrintWithDialog**

Display Windows dialog for sending print-outs.

Prototype:

```
void PrintWithDialog();
```

Parameters:

[None]

Return value:

[None]

### **UnLockActiveX**

Unlock the ActiveX using license key received from Foxit Corporation.

Prototype:

```
void UnLockActiveX(BSTR lisenca_id, BSTR unlock_code)
```

Parameters:

license\_id - A string received from Foxit identifying the SDK licensee

unlock\_code - A string received from Foxit to unlock the ActiveX

Return value:

[None]

Comment:

For evaluating ActiveX, you don't need to call this function and the evaluation marks will be shown on all rendered pages.

For paid ActiveX, you should call this function before calling any other ActiveX functions.

## **UnLockActiveXEx**

Unlock the Activex using license key received from Foxit Corporation.

Prototype:

```
void UnLockActiveXEx(BSTR strLicense)
```

Parameters:

strLicense - A string received from Foxit identifying the SDK license key

Return value:

[None]

Comment:

This function performs the same function as the UnlockActiveX.

## **\*SetUserPassword**

Set User Password for the current PDF file

Prototype:

```
BOOL SetUserPassword(LPCTSTR lpszNewValue)
```

Parameters:

lpszNewValue - Password string

Return value:

Returns True if successful, False otherwise.

## **\*SetUserPermission**

Set User Permission for the current PDF file.

Prototype:

```
BOOL SetUserPermission(long dwPermission)
```

Parameters:

dwPermission - returned User Permission flag.

Return value:

Returns True if successful, False otherwise.

## **\*SetOwnerPassword**

Set Owner Password for the current PDF file

Prototype:

```
BOOL SetOwnerPassword(LPCTSTR lpszNewValue)
```

Parameters:

lpszNewValue - Password string

Return value:

Returns True if successful, False otherwise.

## **\* CheckOwnerPassword**

Valid whether the Owner password is correct.

Ptototype:

```
BOOL CheckOwnerPassword(BSTR lpszPermPsw)
```

Parameters:

lpszPermPsw The owner password

Return value:

Returns True if correct, False otherwise.

### **\*SetContextMenuString**

Set the string for right click contextual menu

Prototype:

Void SetContextMenuString(LPCTSTR string)

Parameters:

string - String containing the right click contextual menu data, such as "a,b,c,d,e". Call this function together with the OnContextMenuIndex event. "a, b, c, d, e" represent distinct menu items.

Return value:

[None]

### **SetPDFMeasureUnit**

Set the measurement unit of PDF documents

Prototype:

BOOL SetPDFMeasureUnit(short nType);

Parameters:

nType - Measurement unit: 0 = Point; 1 = Inch; 2 = Centimeter; 3: Pixel

Return value:

Returns True if successful, False otherwise.

### **GetCurrentWnd**

Obtain the pointer to the current window

Prototype:

Long GetCurrentWnd();

Parameters:

[None]

Return value:

Returns the window point on success, NULL otherwise.

### **GetCtrlInstance**

Obtain the handler to the Control Instance

Prototype:

Long GetCtrlInstance ();

Parameters:

[None]

Return value:

Returns the Control Instance handler on success, NULL otherwise.

### **\*RemoveEvaluationMark**

Remove the Foxit evaluation mark if you have the license key file

Prototype:

BOOL RemoveEvaluationMark()

Parameters:

[None]

Return value:

Returns TRUE if successful, FALSE for failed.

## Events

### BeforeDraw

Triggered Before the painting of the viewer contents is about to begin.

Prototype:

Void BeforeDraw (long dc)

Parameters:

dc - Handle to a device context.

### AfterDraw

Triggered After the painting of the viewer contents is completed.

Prototype:

Void AfterDraw (long dc)

Parameters:

dc - Handle to a device context.

### OnZoomChange

Triggered when you change the Zoomlevel property.

Prototype:

Void OnZoomChange ()

Parameters:

[None]

### OnPageChange

Triggered when you change a page (move from one page to another).

Prototype:

Void OnPageChange ()

Parameters:

[None]

### OnOpenPassword

Triggered when you try to open a PDF document which is password protected.

Prototype:

Void OnOpenPassword (BSTR\* password, BOOL\* cancel)

Parameters:

- Password - Password for the PDF.
- Cancel - When Cancel set False, It will trigger all the time, until password is correct.

### \*OnHyperLink

Triggered when clicking on a hypertext,

Prototype:

```
Void OnHyperLink(BSTR linktype, BSTR linkdata, Link_Dest* dest, BOOL* cancel)
```

Parameters:

- Linktype - A string containing information about the type of hyperlink.

linktype sting are:

GoTo moves to a different page on the current document, the linkdata is null string, dest contain position information which the control is about to navigate.

GoToR moves to a different PDF file stored on the local disk, if the new window is required for viewing the new document, the linkdata information contains the filename followed 1, otherwise, followed 0. dest contain position information which the control is about to navigate.

Launch launches an external application, if the new window is required for viewing the new document, the linkdata information contains the filename followed by 1, otherwise, followed by 0.

URI open an uri, linkdata contains the uri string.

- Cancel - If cancel variable is set to true the control will not follow the hyperlink.

- linkData - A string contains additional information separated by character.

### OnSearchProgress

Triggered when you search document,

Prototype:

```
Void OnSearchProgress (long pageNumber, long pageCount)
```

Parameters:

- pageNumber - The page is currently being searched,
- pageCount - The total number of pages .

### OnOpenFile

Event triggered when file open operation fails

Prototype:

```
void OnOpenFile(short Error);
```

Parameters:

- Error - returns the error code

### **OnFilePathInvalidate**

Event triggered when file operation fails to validate

Prototype:

```
void OnFilePathInvalidate(BSTR WarnString);
```

Parameters:

WarnString - returns the error message

### **OnShowSavePrompt**

Event triggered when closing the modified file

Prototype:

```
void OnShowSavePrompt(BOOL* bShow, short * nResult);
```

Parameters:

bShow - the value indicate whether to show the default message box in ActiveX

nResult - the value indicate whether to save the modified file

### **OnOpenDocument**

Triggered when you open a document.

Prototype:

```
Void OnOpenDocument (BSTR filepath)
```

Parameters:

Filepath - Path to the PDF file.

### **OnCloseDocument**

Triggered when you close a document.

Prototype:

```
Void OnCloseDocument (BSTR filepath)
```

Parameters:

Filepath - Path to the PDF file.

### **OnDocumentChange**

Triggered when the PDF document content change.

Prototype:

```
Void OnDocumentChange ()
```

Parameters:

[None]

### **CustomFileGetSize**

Triggered when using OpenCustomFile method to open PDF document.

Prototype:

```
Void CustomFileGetSize (long* size)
```

Parameters:

- size - [out] Pointer to number that will receive the PDF length. Set it to PDF file length.

### CustomFileGetBlock

Triggered when use OpenCustomFile method to open PDF document.

Prototype:

Void CustomFileGetBlock (long pos, long pBuf, long size)

Parameters:

- pos - [in ] Byte offset from beginning of the file.
- pBuf - [out ]Pointer to the buffer that will receive the pdf data.
- Size - [in] the buffer size.

Comment:

Getting a block of data from specific position. Position is specified by byte offset from beginning of the file. The position and size will never go out range of file length.

### \*OnContextMenuIndex

Event triggered when clicking on an entry on the right click contextual menu. Use this event together with the SetContextMenuString interface.

Prototype:

void OnContextMenuIndex(short nIndex);

Parameters:

- nIndex - Index of the selected menu entry

### OnClick

Triggered when the left button is clicked.

Prototype:

Void OnClick (long hWnd, long ClientX, long ClientY);

Parameters:

- hWnd - The handle of this window.
- ClientX - X coordinate in the ActiveX control window's client area.
- ClientY - Y coordinate in the ActiveX control window's client area.

### OnDbClick

Triggered when the left button is double clicked.

Prototype:

Void OnDbClick (long hWnd, long ClientX, long ClientY);

Parameters:

- hWnd - The handle of this window.
- ClientX - X coordinate in the ActiveX control window's client area.
- ClientY - Y coordinate in the ActiveX control window's client area.



## OnRButtonClick

Event triggered when the right mouse button is clicked

Prototype:

```
void OnRButtonClick(long hWnd, long ClientX, long ClientY);
```

Parameters:

hWnd - The handle of this window.  
ClientX - X coordinate in the ActiveX control window's client area.  
ClientY - Y coordinate in the ActiveX control window's client area.

## OnDownloadFinish

Event triggered when downloading from the Internet succeeds.

Prototype:

```
void OnDownloadFinish();
```

Parameters:

[None]

## OnErrorOccurred

Event triggered when errors occur on the called SDK interfaces. Only methods GetToolByIndex and ShowToolBarButton are supported now.

Prototype:

```
void OnErrorOccurred(BSTR lpszErrorMsg)
```

Parameters:

lpszErrorMsg - The error string

Return value:

[None]

## #FormFieldError

Event triggered when an error occurs while configuring a PDF form field.

Prototype:

```
Void FormFieldError(long nErrorCode);
```

Parameters:

nErrorCode - returns the error code when configuring the PDF form field.

## OnUploadFinish

Event triggered when errors occur on the called UploadCurFileToFTP.

Prototype:

```
void OnUploadFinish(short nRetCode)
```

Parameters:

nRetCode - returns the error code

Return value:

[None]

## OnTextHyperLink

Event triggered when user clicks the text link in ActiveX. It will return the link text to user and let the user to determine whether to trigger this link or not.

Prototype:

```
void OnTextHyperLink(BSTR csUrl, boolean* cancel)
```

Parameters:

csUrl - The url of the text link  
cancel - whether to behavior this link

Return value:

[None]

## \*OnAddMenuItemAction

Event triggered when user adds an “add a menu item” action.

Prototype:

```
void OnAddMenuItemAction(BSTR* pItem)
```

Parameters:

pMenuItem - The menu string defined outside the class

Return value:

[None]

## OnExcuteMenuItem

Event triggered when the user executes an action of the customized menu item that is added by the event OnAddmenuItemAction.

Prototype:

```
void OnExcuteMenuItem(BSTR sMenuItem, boolean* bResult)
```

Parameters:

sMenuItem - customized menu item  
bResult - obtains the result caused by the action (TRUE or FALSE)

Return value:

[None]

## OnDoGoToAction

Event triggered when the user executes the action GoToR.

Prototype:

```
void OnDoGoToAction(BSTR sFilePath, Link_Dest* dest)
```

Parameters:

sFilePath - The targeted file path  
dest - The targeted object in the file

Return value:

[None]

## #PDFForm

### Method

#### AddField

Add a new form field

Prototype:

```
LPDISPATCH AddField(LPCTSTR bstrFieldName, LPCTSTR bstrFieldType,
                    long pageIndex, float left, float top, float right, float bottom);
```

Parameters:

- bstrFieldName - The fully-qualified name of the field.
- bstrFieldType - Field type for the newly created field.

Valid types are:

- text
- button
- combobox
- listbox
- checkbox
- radio button

- pageIndex - The page number (zero-based).
- left - The left coordinates of the field rectangle.
- top - The top coordinates of the field rectangle.
- right - The right coordinates of the field rectangle.
- bottom - The bottom coordinates of the field rectangle.

Return value:

The newly-created field object.

Comment:

The field rectangle coordinate measured in rotated page space; that is, [0,0] is always at the left bottom corner regardless of page rotation.

#### RemoveFieldsByName

delete a specific form field

Prototype:

```
Void RemoveFieldsByName (LPCTSTR bstrFieldName);
```

Parameters:

- bstrFieldName - The fully-qualified name of the field to be removed. If the field has multiple child annotations, all of them are removed. If multiple fields have the same name, all are removed.

Return value:

[None]

## ExportToFDF

Export certain form elements into an FDF file.

Prototype:

```
Void ExportToFDF (LPCTSTR bstrFullPath, BOOL bEmptyFields,  
                  const VARIANT FAR& arrFields)
```

Parameters:

bstrFullPath	-	Full path of the exported FDF file
bEmptyFields	-	True to export ONLY elements specified in VARIANT& arrField. False to export everything but the elements specified in VARIANT& arrFields.
ArrFields	-	Form element array to be exported

Return value:

[None]

## ImportFromFDF

Import from an FDF file

Prototype:

```
void ImportFromFDF(LPCTSTR bstrFullPath);
```

Parameters:

bstrFullPath	-	Full path for the FDF file.
--------------	---	-----------------------------

Return value:

[None]

## GetFieldByIndex

Obtain the pointer for a specific form field

Prototype:

```
LPDISPATCH GetFieldByIndex(long index)
```

Parameters:

index	-	Index of the form field
-------	---	-------------------------

Return value:

On Success, returning the pointer to the form field, Null otherwise.

## GetFieldsCount

Obtain the count of all form fields

Prototype:

```
long GetFieldsCount()
```

Parameters:

[None]

Return value:

On Success, returning the number of form fields, -1 otherwise.

## RemoveField

Remove a specific form field.

Prototype:

```
void RemoveField(CPDFFormField pFormField)
```

Parameters:

pFormField - The input form field to be removed

Return value:

[None]

## #PDFFormField

### Properties

#### Alignment

Type:

String

Description:

Alignment of text inside a textfield (left, center, right)

Comment:

For text field only.

#### BorderStyle

Type:

String

Description:

Style of the form field border

Comment:

For all.

The border style can be set to: 1) solid 2) dashed 3) beveled 4) inset 5) underline

#### BorderWidth

Type:

short

Description:

Width of the form field

Comment:

For all

#### ButtonLayout

Type:

short

Description:

The layout appearance of a button. Valid values include:

0 - Text only; the button has a caption but no icon.

1 - Icon only; the button has an icon but no caption.

2 - Icon over text; the icon should appear on top of the caption.

3 - Text over icon; the text should appear on top of the icon.

4 - Icon then text; the icon should appear to the left of the caption.

5 - Text then icon; the icon should appear to the right of the caption.

6 - Text over icon; the text should be overlaid on top of the icon.

Comment:

For push button only

### **CalcOrderIndex**

Type:

short

Description:

Index of the current form field in the CO array

Comment:

For all

### **CharLimit**

Type:

short

Description:

Limit of the number of characters in a textfield.

Comment:

Text Field Only

### **DefaultValue**

Type:

String

Description:

Default value of the form field

Comment:

For all

### **IsEditable**

Type:

Boolean

Description:

Indicate if the Combo Box is editable

Comment:

Combo Box Only

### **Behavior**

Type:

String

Description:

None, Invert, Outline, Push



N (None)	No highlighting.
I (Invert)	Invert the contents of the annotation rectangle.
O (Outline)	Invert the annotation's border.
P (Push)	Display the annotation as if it were being pushed below the surface of the page.

### **IsHidden**

Type:

Boolean

Description:

Whether the form field is hidden

Comment:

For all

### **IsMultiline**

Type:

Boolean

Description:

Whether a textfield is multi-line or single-line.

Comment:

Text field only

### **IsPassword**

Type:

Boolean

Description:

Whether to mask the input as password input.

Comment:

Text Field only.

### **IsReadOnly**

Type:

Boolean

Description:

Whether to set a form field as read only.

Comment:

For All

### **IsRequired**

Type:

Boolean

**Description:**

Whether a field has to be non-empty

**Comment:**

For Combo box, Radio button, Text Field

**Name****Type:**

String

**Description:**

Name of the current form field

**Comment:**

Read only, For all

**NoViewFlag****Type:**

Boolean

**Description:**

Whether or not to show the form element. 1 = show; 0 = hidden.

**Comment:**

For all

**PrintFlag****Type:**

Boolean

**Description:**

Whether or not to include form element in print-outs. 1 = show; 0 = hidden.

**Comment:**

For all

**Style****Type:**

CString

**Description:**

Set the shape of checkbox and radio button to:

1) check, 2) cross, 3) diamond, 4) circle, 5) star, 6) square

**Comment:**

For checkbox, radio button

**TextFont****Type:**

String

**Description:**

Font (Reference 1.7 416)

Can be set as:

- Courier
- Courier-Bold
- Courier-Oblique
- Courier-BoldOblique
- Helvetica
- Helvetica-Bold
- Helvetica-Oblique
- Helvetica-BoldOblique
- Symbol
- Times-Roman
- Times-Bold
- Times-Italic
- Times-BoldItalic
- ZapfDingbats

**Comment:**

For all except for Check box, radio box

**TextSize****Type:**

Short

**Description:**

Size of the text in form fields

**Comment:**

For all except for Check box, radio box

**Type****Type:**

String

**Description:**

Type of the Form

**Comment:**

For all

It can be set as: text, button, combobox, listbox, checkbox, radiobutton

**Value****Type:**

String

**Description:**

The current value

Comment:

Form elements having this property:

1) Text field 2) combobox 3) radio button 4) checkbox <Yes&Off> 5) Listbox

## Tooltip

Type:

String

Description:

Displayed tooltip

Comment:

For all

## Orientation

Type:

Short

Description:

The text rotation of the form

Comment:

For all

## Method

### PopulateListOrComboBox

Assign values to entries within a Listbox or Combobox

Prototype:

```
Void PopulateListOrComboBox ( const VARIANT& arrItems,  
                             const VARIANT& arrExportVal);
```

Parameter:

arrItem	-	An array of strings, with each element representing an item name.
arrExportVal	-	An array of strings, the same size as the first parameter, with each element representing an export value.

Return Value:

[None]

### SetBackgroundColor

Set the background color of a form field

Prototype:

```
Void SetBackgroundColor (LPCTSTR bstrColorSpace, float redC, float greenM,  
                        float blueY, float AlphaK);
```

Parameter:

- bstrColorSpace - Can be one of the following: transparent, gray, RGB or CMYK color space. Use T, G, RGB and CMYK for each.  
For T and G, redC is required; For RGB, redC, greenM and blueY are required; For CMYK, redC, greenM, blueY and AlphaK are required.

redC, greenM and blueY are values from 0 to 1.

Return value:

[None]

## SetBorderColor

Set the border color.

Prototype:

```
void SetBorderColor (LPCTSTR bstrColorSpace, float redC, float greenM, float blueY, float AlphaK);
```

Parameter:

- bstrColorSpace - Can be one of the following: transparent, gray, RGB or CMYK color space. Use T, G, RGB and CMYK for each.  
For T and G, redC is required; For RGB, redC, greenM and blueY are required; For CMYK, redC, greenM, blueY and AlphaK are required.

redC, greenM and blueY are values from 0 to 1.

Return value:

[None]

## SetForegroundColor

Set the foreground color

Prototype:

```
Void SetForegroundColor (LPCTSTR bstrColorSpace, float redC, float greenM, float blueY, float AlphaK);
```

Parameter:

- bstrColorSpace - Can be one of the following: transparent, gray, RGB or CMYK color space. Use T, G, RGB and CMYK for each.  
For T and G, redC is required; For RGB, redC, greenM and blueY are required; For CMYK, redC, greenM, blueY and AlphaK are required.

redC, greenM and blueY are values from 0 to 1.

Return value:

[None]

## SetButtonCaption

Set the text displayed on a button

Prototype:

```
void SetButtonCaption (LPCTSTR bstrFace, LPCTSTR bstrCaption);
```

Parameter:

**bstrFace** - A string that specifies the face for which the caption will be used.

Valid strings include:

N — Normal appearance

D — Down appearance

R — Appearance for rollover

**bstrCaption** - The caption for the button.

Return value:

[None]

## SetButtonIcon

Set the icon for a button

Prototype:

```
Void SetButtonIcon (LPCTSTR bstrFace, LPCTSTR bstrFilePath);
```

Parameters:

**bstrFace** - A string that specifies the face for which the caption will be used.

Valid strings include:

N — Normal appearance

D — Down appearance

R — Appearance for rollover

**bstrFilePath** - The full path of the image file to be used as the source of the appearance.

Return Value:

[None]

## SetExportValues

When Radio Button and Checkbox are to be exported, set the value to export for different selection (Selected, un-selected, checked, un-checked, etc)

Prototype:

```
void SetExportValues (const VARIANT& arrExportVal);
```

Parameters:

**arrExportVal** - Array of the export values

Return Value:

[None]

## SetJavaScriptAction

Set the action of JavaScript

Ptotype:

```
Void SetJavaScriptAction (LPCTSTR bstrTrigger, LPCTSTR bstrJavaScript);
```

## Parameters:

bstrTrigger - A string that specifies the trigger for the action.

Valid strings include:

- up
- down
- enter
- exit
- calculate
- validate
- format
- keystroke

bstrJavaScript - The script itself.

## Return value:

[None]

### SetResetFormAction

Set the action of the Reset form field

## Prototype:

```
void SetResetFormAction (LPCTSTR bstrTrigger, long bFlags, const VARIANT&
arrFields);
```

## Parameters:

bstrTrigger - A string that specifies which trigger is used for the action.

Valid strings include:

- up — Mouse up
- down — Mouse down
- enter — Mouse enter
- exit — Mouse exit

bFlags - A collection of flags that define various characteristics of the action.

arrFields - Form element array to be exported

## Return value:

[None]

### SetSubmitFormAction

Set the action for the Submit form field

## Prototype:

```
Void SetSubmitFormAction (LPCTSTR bstrTrigger, LPCTSTR bstrURL, long
bFlags, const VARIANT& arrFields);
```

## Parameters:

bstrTrigger - A string that specifies which trigger is used for the action.

Valid strings include:

- up - Mouse up
- down - Mouse down
- enter - Mouse enter

exit - Mouse exit

- bstrURL - A string containing the URL.
- bFlags - A collection of flags that define various characteristics of the action.
- arrFields - Form element array to be submitted

Return value:

[None]

### **GetPageIndex**

Obtain the page index of specified field.

Prototype:

long GetPageIndex()

Parameters:

[None]

Return value:

Returns the page index of the field

### **GetRectTop**

Obtain the top of specified field.

Prototype:

float GetRectTop(I)

Parameters:

[None]

Return value:

Returns the top of the field

### **GetRectLeft**

Obtain the left of specified field.

Prototype:

float GetRectLeft()

Parameters:

[None]

Return value:

Returns the left of the field

### **GetRectRight**

Obtain the right of specified field.

Prototype:

float GetRectRight()

Parameters:

[None]

Return value:



Returns the right of the field

### **GetRectBottom**

Obtain the bottom of specified field.

Prototype:

```
float GetRectBottom()
```

Parameters:

[None]

Return value:

Returns the bottom of the field

## **^ CPDFPageAnnots**

### **Method**

#### **GetPageAnnots**

Obtain a reference to the annotations in a specific PDF page.

Prototype:

```
CPDFPageAnnots GetPageAnnots(long pageIndex)
```

Parameters:

pageIndex            The specific page number where to obtain annotations.

Return Value:

Returns a CPDFPageAnnots object that contains references to all of the annotations in the page if successful, otherwise NULL .

#### **GetAnnot**

Obtain a pointer to a specific PDF annotation.

Prototype:

```
CPDFAnnot GetAnnot(long AnnotIndex)
```

Parameters:

AnnotIndex            Index of the PDF annotation.

Return Value:

Return a pointer to the specific PDF annotation if successful, otherwise NULL.

#### **AddAnnot**

Add a new PDF annotation

Prototype:

```
CPDFAnnot AddAnnot (LPDISPATCH AnnotToAddAfter, LPCTSTR SubType, float left, float top, float right, float bottom)
```

**Parameters:**

AnnotToAddAfter	Reserved. Set to NULL.
SubType	Subtype of the annotation. Possible types: "Cloudy", "Arrow", "Line", "Square", "Rectangle", "Circle", "Ellipse", "Polygon", "PolyLine", "Pencil", "Underline", "Highlight", "Squiggly", "StrikeOut", "Replace", "Caret", "Note", "Typewriter", "Callout", "Textbox", "FileAttachment", "Image", "Movie", "Sound", "Rectangle Link", "Quadrilateral Link".
left	The left coordinates of the annotation rectangle.
top	The top coordinates of the annotation rectangle.
right	The right coordinates of the annotation rectangle.
bottom	The bottom coordinates of the annotation rectangle.

All coordinates are based on the PDF coordinate space.

**Return Value:**

The new annotation object if successful, otherwise will be NULL.

**RemoveAnnot**

Delete a specific PDF annotation.

**Prototype:**

```
long RemoveAnnot(LPDISPATCH AnnotToRemove)
```

**Parameters:**

AnnotToRemove	A reference to the specific PDF annotation that to be removed.
---------------	--

**Return Value:**

Returns 0 if successful, otherwise -1.

**GetAnnotIndex**

Obtain the index of the specific PDF annotation.

**Prototype:**

```
long GetAnnotIndex(LPDISPATCH Annot)
```

**Parameters:**

Annot	A reference to the specific PDF annotation
-------	--

**Return Value:**

Returns the index of the specific PDF annotation if successful, otherwise will be -1.

**GetAnnotsCount**

Obtain the count of all PDF annotations in the current PDF page.

**Prototype:**

```
long GetAnnotsCount()
```

**Parameters:**

NULL

**Return Value:**

Returns the number of PDF annotations.

## **^CPDFAnnot**

### **Properties**

#### **Thickness**

**Type:**

short. Read and Write.

**Description:**

The width of PDF annotation border.

**Comment:**

Used by Drawing Annotations, Measure Tools, and any annotations that have a border (i.e. Cloudy, Arrow, Line, Square, Rectangle, Circle, Ellipse, Polygon, Polyline, Pencil, Callout, Textbox, Link).

#### **BorderStyle**

**Type**

Short. Read and Write

**Description**

Style of the annotation border.

**Comment**

There are 7 types:

- \* 1 Solid For all types of annotations.
- \* 2 Dashed type 1 For Drawing Annotations only.
- \* 3 Dashed type 2 For Drawing Annotations only.
- \* 4 Dashed type 3 For Drawing Annotations only.
- \* 5 Dashed type 4 For Drawing Annotations only.
- \* 6 Cloudy type 1 For these Annotations only: Cloudy, Polygon, Circle, Square, Rectangle.
- \* 7 Cloudy type 2 For these Annotations only: Cloudy, Polygon, Circle, Square, Rectangle.

Annotations that aren't supported: Pencil, Underline, Highlight, Squiggly, StrikeOut, Replace, Caret Note, Typewriter, FileAttachment, Image, Movie, and Sound.

#### **Color**

**Type**

OLE\_COLOR. Read and Write.

Description

The background color of PDF annotation.

Comment

Used by all annotations.

## LineStartingStyle

Type

Short. Read and Write.

Description

The starting style of line.

Comment

Used by the following annotations: Line, Arrow, PolyLine and Callout

There are 10 types:

- \* 0 None
- \* 1 Square
- \* 2 Round
- \* 3 Diamond
- \* 4 Open
- \* 5 Close
- \* 6 Butt
- \* 7 Open(recersed)
- \* 8 Close(recersed)
- \* 9 Slash

## LineEndingStyle

Type

Short. Read and Write.

Description

The end style of a line.

Comment

Used by the following annotations: Line, Arrow, and PolyLine.

There are 10 types:

- \* 0 None
- \* 1 Square
- \* 2 Round
- \* 3 Diamond
- \* 4 Open
- \* 5 Close
- \* 6 Butt
- \* 7 Open(recersed)
- \* 8 Close(recersed)

\* 9 Slash

## FillColor

Type

OLE\_COLOR. Read and Write.

Description

The fill color of PDF annotation

Comment

Used by the following annotations: Cloudy, Arrow, Line, Square, Rectangle, Circle, Ellipse, Polygon, and Polyline.

## Opacity

Type

Short. Read and Write.

Description

The opacity value of the PDF annotation.

Comment

Annotations not supported: Link, Movie, and Sound.

## Author

Type

BSTR. Read and Write.

Description

The author of the PDF annotation.

Comment

Annotations not supported: Image, Movie, Sound, and Link.

## Subject

Type

BSTR. Read and Write.

Description

The subject of this PDF annotation

Comment

Annotations not supported: Image, Movie, Sound, and Link.

## CreationDate

Type

DATE. Read only.

Description

The create time of this PDF annotation.

## **ModificationDate**

### Type

DATE. Read only.

### Description

The date of the last modification made to the PDF annotation.

## **Locked**

### Type

Boolean. Read and Write.

### Description

Flag indicating whether the annotation is locked.

### Comment

Used by all annotations.

## **Print**

### Type

Boolean. Read and Write.

### Description

Flag indicating whether the annotation can be printed.

### Comment

Used by all annotations.

## **ReadOnly**

### Type

Boolean. Read and Write.

### Description

Flag indicating whether the annotation is read only.

### comment

Used by all annotations.

## **Description**

### Type

BSTR. Read and Write.

### Description

The description of the PDF annotation

### Comment

Only can be used for FileAttachment Annotation.

## Methods

### GetType

Obtain the type of PDF annotation

Prototype:

BSTR GetType()

Parameters:

None

Return Value:

Return the type of PDF annotation

### GetSubType

Obtain the subtype of PDF annotation

Prototype:

BSTR GetSubType()

Parameters:

None

Return Value:

Return the Subtype of the PDF annotation

### GetContents

Obtain the contents of PDF annotation

Prototype:

BSTR GetContents()

Parameters:

None

Return Value:

Return the contents of the PDF annotation

### SetContents

Set the contents of the PDF annotation

Prototype:

long SetContents(LPCTSTR Contents)

Parameters:

Contents            The contents to be set.

Return Value:

Return 0 if successful, otherwise -1

### IsPopupOpen

Indicates whether the popup box is opened or not.

Prototype:

BOOL IsPopupOpened()

Parameters:

None

Return Value:

Returns TRUE if the Popup Box is open, otherwise FALSE.

## SetPopupOpen

Set whether to open or close the Popup Box.

Prototype:

long SetPopupOpened(BOOL Open)

Parameters:

Open            The input value to set whether to open the Popup Box.

Return Value:

Returns 0 if successful, otherwise -1.

## HasPopup

Indicates whether the annotation is a Markup Annotation(has a Popup Box)

Prototype:

BOOL HasPopup()

Parameters:

None

Return Value:

Returns TRUE if the Annotation has the Popup Box, otherwise FALSE.

## GetRect

Get the rectangle of the annotation.

Prototype:

long GetRect(float\* pLeft, float\* pTop, float\* pRight, float\* pBottom)

Parameters:

pLeft            The output value for receiving the left value of the rect.

pTop             The output value for receiving the top value of the rect.

pRight           The output value for receiving the right value of the rect.

pBottom          The output value for receiving the bottom value of the rect.

Return Value:

Return 0 if successful, otherwise -1.

## SetRect

Set the rectangle of the annotation.

Prototype:

long SetRect(float Left, float Top, float Right, float Bottom)



**Parameters:**

Left	The input value for setting the left value of the rect.
Top	The input value for setting the top value of the rect.
Right	The input value for setting the right value of the rect.
Bottom	The input value for setting the bottom value of the rect.

**Return Value:**

Return 0 if successful, otherwise -1.

**SetLinkGoToAction**

Set the GoTo Action for a link annotation.

**Prototype**

```
void SetLinkGoToAction(long nPageIndex, float left, float top, float zoom)
```

**Parameters**

nPageIndex	The input page number.
left,top	The position of the windows' upper-left corner..
zoom	The magnified factor of page.

**Return Value**

None

**SetLinkURLAction**

Set the URL Action of the link annotation.

**Prototype:**

```
void SetLinkURLAction(LPCTSTR sURL)
```

**Parameters:**

sURL	The uniform resource identifier to go to when the action is executed.
------	---

**Return Value:**

None

**DoAction**

Perform the link annotation's action, if it has.

**Prototype:**

```
long DoAction()
```

**Parameters:**

None

**Return Value:**

Returns 0 if successful, otherwise -1.

**HasAction**

Indicates whether the link annotation has an action.

**Prototype:**

BOOL HasAction()

Parameters:

None

Return Value:

Returns TRUE if the link annotation has an action, otherwise FALSE.

### **GetMarkedState**

Obtain the mark state of this annotation.

Prototype

long GetMarkedState()

Parameters

None

Return Value

The return value could be 0 (unmarked), 1 (marked), or -1 indicating an error.

### **SetMarkedState**

Set the mark state of this annotation. annotations not supported:Image, Movie, Sound, and Link.

Prototype:

long SetMarkedState(long state)

Parameters:

state           The input value to set the mark state of the annotation. Possible values are 0 (unmarked) or 1 (marked).

Return Value:

Return 0 if successful, otherwise -1.

### **GetReviewState**

Get the review state of this annotation.

Prototype:

long GetReviewState()

Parameters:

None

Return Value:

The return value could be 0, 1, 2, 3, 4, which mean None, Accepted, Rejected, Canceled, and Completed, respectively. Otherwise -1 to indicate an error.

### **SetReviewState**

Set the review state of this annotation. Annotations not supported:Image, Movie, Sound, and Link.

Prototype:

long SetReviewState(long state)

Parameters:

state            The input value to set the review state of the annotation. Possible values are 0, 1, 2, 3, 4, which means None, Accepted, Rejected, Canceled, and Completed.

Return Value:

Return 0 if successful, otherwise -1.

### **GetMigrationState**

Get the migration state of this annotation.

Prototype:

```
long GetMigrationState()
```

Parameters:

None

Return Value:

The return value can be 0, 1, or 2, which mean None, Not Confirmed, and Confirmed respectively. Otherwise -1.

### **SetMigrationState**

Set the migration state of this annotation. Annotations not supported :Image, Movie, Sound, and Link.

Prototype:

```
long SetMigrationState(long state)
```

Parameters:

state            The input value to set the migration state of the annotation. Possible values are 0, 1, or 2, which mean None, Not Confirmed, and Confirmed respectively.

Return Value:

Return 0 if successful, otherwise -1.

### **SetStartingPoint**

For Line and Arrow annotations. Set the start point of the annotation.

Prototype:

```
long SetStartingPoint(float PointX, float PointY)
```

Parameters:

PointX          The input value to set the x of the start point.

PointY          The input value to set the y of the start point.

Return Value:

Return 0 if successful, otherwise -1.

### **GetStartingPoint**

For Line and Arrow annotations. Get the start point of the annotation.

Prototype:

long GetStartingPoint(float\* PointX, float\* PointY)

Parameters:

PointX        The output value for receiving the x of the start point.

PointY        The output value for receiving the y of the start point.

Return Value:

Return 0 if successful, otherwise -1.

### **SetEndingPoint**

For Line and Arrow annotations. Set the end point of the annotation.

Prototype:

long SetEndingPoint(float PointX, float PointY)

Parameters:

PointX        The input value to set the x of the end point.

PointY        The input value to set the y of the end point.

Return Value:

Return 0 if successful, otherwise -1

### **GetEndingPoint**

For Line and Arrow annotations. Get the end point of the annotation.

Prototype:

long GetEndingPoint(float\* PointX, float\* PointY)

Parameters:

PointX        The output value for receiving the x of the end point.

PointY        The output value for receiving the y of the end point.

Return Value:

Return 0 if successful, otherwise -1.

### **SetMediaPoster**

Set a poster (image) for annotations that support posters (i.e. Movie and Sound annotations)

Prototype:

long SetMediaPoster(LPCTSTR ImageFilePath)

Parameters:

ImageFilePath        The input image file path.

Return Value:

Return 0 if successful, otherwise -1.

### **SetMultimedia**

Set the multimedia content for annotations that support multimedia.

Prototype:

long SetMultimedia(LPCTSTR FilePath, LPCTSTR ContentType, BOOL Embed,

BOOL bShowCtrlBar)

Parameters:

FilePath	The media file path.
ContentType	The MIME type of the media data.
Embed	Indicates whether to embed media in the PDF file.
bShowCtrlBar	Indicates whether to show the control bar. .

Return Value:

Returns 0 if successful, otherwise -1.

### **SetLinkQuadPoints**

Set Link annotation's appearance.

Prototype:

long SetLinkQuadPoints(long\* PointsArray, long PointsCount)

Parameters:

PointsArray	An array of points.
PointsCount	The size of the array must be 4.

Return Value:

Returns 0 if successful, otherwise -1.

### **SetPolygonVertices**

Set Polygon annotation's appearance.

Prototype:

long SetPolygonVertices(long\* PointsArray, long PointsCount)

Parameters:

PointsArray	An array of points.
PointsCount	The size of PointArray.

Return Value:

Returns 0 if successful, otherwise -1.

### **SetPencilVertices**

Set Pencil annotation's appearance.

Prototype:

long SetPencilVertices(long\* PointsArray, long PointsCount)

Parameters:

PointsArray	An array of lines, each line is a point array.
PointCount	The size of PointArray.

Return Value:

Returns 0 if successful, otherwise -1.

### **AttachFile**

Attach a file to a FileAttachment annotation.

Prototype:

long AttachFile(LPCTSTR FileName)

Parameters:

FileName            The input file path.

Return Value:

Returns 0 if successful, otherwise -1.

## **^CPDFormatTool**

CPDFormatTool is an interface that allows users to format free text annotations (FreeTextAnnot) in PDF applications. The Interface allows a developer to utilize the same formatting tools found in the Foxit Reader annotation toolbar. There are three FreeTextAnnots in ActiveX, Typewriter, Callout, and TextBox. Each FreeTextAnnot maintains independent format data. Please note that when you change the current FreeTextAnnot, the data of CPDFormatTool will also be modified. CPDFormatTool works only for formatting FreeTextAnnot objects and not any other type of annotations.

### **Method**

#### **GetFormatTool**

Obtain a reference to a FormatTool.

Prototype:

CPDFormatTool GetFormatTool ()

Parameters:

None

Return Value:

Returns a reference to a FormatTool if successful, otherwise NULL.

#### **SetFontName**

Set current font.

Prototype:

Void SetFontName(BSTR FontName)

Parameters:

FontName            The input font name.

Return Value:

None

#### **GetFontName**

Get current font.

Prototype:

BSTR GetFontName()

Parameters:

None

Return Value:

The current font name.

### **SetFontSize**

Set the font size.

Prototype:

Void SetFontSize(float FontSize)

Parameters:

FontSize      The input font size.

Return Value:

None

### **GetFontSize**

Get current font size.

Prototype:

Float GetFontSize.

Parameters:

None

Return Value:

Return current font size.

### **SetFontColor**

Set current text color.

Prototype:

Void SetFontColor(OLE\_COLOR FontColor)

Parameters:

FontColor      The input font color.

Return Value:

None

### **GetFontColor**

Obtain the current font color.

Prototype:

OLE\_COLOR GetFontColor()

Parameters:

None

Return Value:

Returns the current font color.

## SetBorderColor

Set the border color.

Prototype:

```
Void SetBorderColor(OLE_COLOR color)
```

Parameters:

Color            The input border color.

Return Value:

None

## GetBorderColor

Obtain the border color.

Prototype:

```
OLE_COLOR GetBorderColor()
```

Parameters:

None

Return Value:

Returns the border color.

## SetFillColor

Set fill color.

Prototype:

```
Void SetFillColor(OLE_COLOR FillColor)
```

Parameters:

FillColor        The input fill color to be set.

Return Value:

None

## GetFillColor

Get fill color.

Prototype:

```
OLE_COLOR GetFillColor()
```

Parameters:

None

Return Value:

Return fill color.

## SetFontBold

Set whether to display the text in bold .

Prototype:

```
Void SetFontBold(BOOL FontBold)
```

Parameters:



**FontBold**                      The input value to be set whether to display the text in bold .

Return Value:

None

Note:

The bold and italic properties currently can be set when the font is one of the following:

Courier\Helvetica\Times Roman.

### **GetFontBold**

Indicates if the text is bold.

Prototype:

BOOL GetFontBold()

Parameters:

None

Return Value:

Returns TRUE if the current is bold , otherwise FALSE.

### **GetFontBoldEnable**

Indicates if bold can be enabled by the format tool.

Prototype:

BOOL GetFontBoldEnable()

Parameters:

None

Return Value:

Returns TRUE if bold can be enabled, otherwise FALSE.

### **SetFontItalic**

Set whether to display the text in italic.

Prototype:

Void SetFontItalic(BOOL FontItalic)

Parameters:

FontItalic      TRUE to display text in italic, otherwise FALSE.

Return Value:

None

### **GetFontItalic**

Indicates if the text is italic.

Prototype:

BOOL GetFontItalic()

Parameters:

None

**Return Value:**

Returns TRUE if the text is italic, otherwise FALSE.

**GetFontItalicEnable**

Indicates if the font italic can be set.

**Prototype:**

```
BOOL GetFontItalicEnable()
```

**Parameters:**

None

**Return Value:**

Returns TRUE if font italic is available, otherwise FALSE.

**SetAlign**

Set the alignment of text.

**Prototype:**

```
Void SetAlign(AlignStyle Style)
```

**Parameters:**

Style	The input alignment style. Can be set as
	ASLEFT =0,
	ASMIDDLE =1,
	ASRIGHT =2

**Return Value:**

None

**GetAlign**

Get the alignment style of the text.

**Prototype:**

```
AlignStyle GetAlign()
```

**Parameters:**

None

**Return Value:**

Returns the alignment style of the text. Possible values are:

ASLEFT	=0,
ASMIDDLE	=1,
ASRIGHT	=2

**SetCharSpace**

Set the spacing between characters.

**Prototype:**

```
Void SetCharSpace(float CharSpace)
```

**Parameters:**

CharSpace                      Space between characters.  
Return Value:  
    None

### **GetCharSpace**

    Get the spacing between characters.  
Prototype:  
    float GetCharSpace()  
Parameters:  
    None  
Return Value:  
    Returns the space between characters.

### **SetCharHorzScale**

    Set the horizontal scale of characters.  
Prototype:  
    Void SetCharHorzScale(float CharHorzScale)  
Parameters:  
    CharHorzScale      The input scale to be set.  
Return Value:  
    None

### **GetCharHorzScale**

    Get the horizontal scale of characters.  
Prototype:  
    float GetCharHorzScale()  
Parameters:  
    None  
Return Value:  
    Returns the horizontal scale.

## PDFPrinter

With IPDFPrinter interface you can control the printer and send print-outs.

### Properties

#### PrinterName

Type:

BSTR

Description:

Set the printer name that will be used for print-outs.

#### printerRangeMode

Type:

PrinterRangeMode

Description:

Set the printer name that will be used for print-outs.

Set the print range, can be set to :

```
PRINT_RANGE_ALL           = 0,  
PRINT_RANGE_CURRENT_VIEW  = 1,  
PRINT_RANGE_CURRENT_PAGE  = 2,  
PRINT_RANGE_SELECTED      = 3,
```

#### printerRangeFrom

Type:

short

Description:

Specify the first page number to be printed.

You must first set the PrinterRangeMode to PRINT\_RANGE\_SELECTED

## **printerRangeTo**

Type:

short

Description:

Specify the last page number to be printed.

You must first set the PrinterRangeMode to PRINT\_RANGE\_SELECTED

## **numOfCopies**

Type:

short

Description:

Specify the number of copies to be printed.

## **Scaling**

Type:

short

Description:

Set the scaling of in the print dialog.

## **AutoRotate**

Type:

boolean

Description:

Set the auto-rotate parameter in the print dialog

## **AutoCenter**

Type:

boolean

Description:

Set whether or not to auto-center when printing. 1 = auto-center; 0 = no auto-center

## **Collate**

Type:

boolean

Description:

Set whether or not to check the Collate option in print dialog. 1 = Collate checked; 0 = not checked.

## Rotation

Type:

short

Description:

Set whether or not to rotate the document when printing.

## RangeSubset

Type:

short

Description:

Set whether or not to include Subset when printing.

## ReversePage

Type:

boolean

Description:

Set whether or not to print document in reverse order.

## PageBorder

Type:

boolean

Description:

Set whether or not to print out page borders.

## Methods

### PrintWithDialog

Display Windows dialog for sending print-outs.

Prototype:

```
void PrintWithDialog();
```

Parameters:

[None]

Return value:

[None]

### PrintQuiet

Send the printouts to the specified printer without displaying the print dialog.

Prototype:

```
void PrintQuiet();
```

Parameters:

[None]

Return value:

[None]

## **SetPaperSize**

Set the paper size for the selected printer, for available paper sizes values print please read Windows SDK documentation.

Prototype:

```
void SetPaperSize (long paperSize);
```

Parameters:

paperSize - Available paper sizes.

Return value:

[None]

## PDFOutline

Give the PDF outline object information.

### Methods

#### NavigateOutline

Navigate to the destination specified by the outline object.

Prototype:

```
void    NavigateOutline ();
```

Parameters:

[None]

Return value:

[None]

#### GetOutlineTitle

Get the title of the outline object.

Prototype:

```
BSTR    GetOutlineTitle()
```

Parameters:

[None]

Return value:

Return the title of the outline object.

#### GetOutLineTitle2

Get the title of the outline object as a variant.

Prototype:

```
VARIANT    GetOutLineTitle2 ()
```

Parameters:

[None]

Return value:

Return the title of the outline object as a variant



## PDFDocumentInfo

Information regarding the document properties.

### Properties

#### Author

Type:

BSTR

Description:

Author of the PDF document.

#### Subject

Type:

BSTR

Description:

Subject of the PDF document.

#### CreatedDate

Type:

BSTR

Description:

Creation date of the PDF document.

#### ModifiedDate

Type:

BSTR

Description:

Modified date of the PDF document.

#### Keywords

Type:

BSTR

Description:

Keywords for the PDF document.

#### Creator

Type:

BSTR

Description:

Creator of the PDF document.

## **Producer**

Type:

BSTR

Description:

Producer of the PDF document.

## **Title**

Type:

BSTR

Description:

Title of the PDF document.

## FindResult

The FindResult class represents a search result if a search is performed and an occurrence is found.

### Methods

#### GetFindPageNum

Get the page index of the search result.

Prototype:

```
long GetFindPageNum();
```

Parameters:

[None]

Return value:

The page index.

#### GetFindFileName

Get the find file name of the search result.

Prototype:

```
BSTR GetFindFileName();
```

Parameters:

[None]

Return value:

The file name.

#### GetFindString

Get the context of the search result.

Prototype

```
BSTR GetFindString()
```

Parameters:

[None]

Return Value

Return the context of the search result.