

# **Foxit Reader ActiveX SDK 2.4**

## **Programming Guide**



©2008 Foxit Software Company

All rights reserved.

2008.12

## Overview

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

Foxit Reader ActiveX SDK 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 rich features built inside it. A programmer can simply drag and drop the component into their application and instantly add PDF displaying functionality to the application. The ActiveX allows users to navigate, zoom, rotate, scroll and print out PDF documents.

Version 2.4 incorporates advanced PDF features. It supports annotation and allows users to fill out, import or export PDF forms. Version 2.4 exposes more functions and events, giving programmers flexible control over the component and more access to the PDF documents.

Foxit offers two versions of ActiveX 2.4. One is the standard version that offers all but the following features: creating/editing annotation, importing/exporting form data, running JavaScript, converting PDF to text, etc. The other one is the professional version that includes all the features. You may choose the version that's right for you based on the requirements of your application. For standard version, you may simply purchase it online. For professional version, you need to contact [sales@foxitsoftware.com](mailto:sales@foxitsoftware.com) to discuss licensing details. In this developer's guide, we mark all properties and functions that are available only in professional version with an asterisk (\*).

Foxit Reader ActiveX SDK runs on Windows 95/NT or later OS. It is a stand-alone component and does not require extra PDF software such as Foxit Reader installed. A user might need to have the administrator's right in order to successfully register the ActiveX under Windows.

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. You may download them from website [www.foxitsoftware.com](http://www.foxitsoftware.com).

**UNLOCK Code:** If you have purchased Foxit Reader ActiveX SDK and received the full version of the ActiveX and the unlock code, you should call [UnLockActiveX](#) function 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.

## Tutorials

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

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

In the following example, suppose we have 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,  
// otherwise, 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, and then a print dialog will pop up and user 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](#), [GetOutlineNextSibling](#) to iterate whole outline tree, then you can get the each 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 return nonzero value and the occurrence will be highlighted. And then you may call [FindNext](#) to search for the next occurrence.

If you want to search text inside PDF files without opening and displaying them, you may use [FindFileFirst](#) and [FindFileNext](#).

## 9) Annotations (\*)

End users may 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”.

## FoxitReaderSDK 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.

### Properties:

#### FilePath

**Type:**

BSTR, read-only

**Description:**

Full path to the current opened PDF file. If no PDF file is opened, the property is an empty string. By using function **GetFilePath**, you can get the value of FilePath (this function can only be used in VC).

#### Password

**Type:**

BSTR, read-only

**Description:**

Password for the PDF. By calling function **GetPassword**, you can get the password of this document (this function can only be used in VC).

#### PageCount

**Type:**

long, read-only

**Description:**

Total number of pages in the current opened PDF file. By calling **GetPageCount** method, you can get the total number of pages in current document (this function can only be used in VC).

#### CurPage

**Type:**

long, read-only

**Description:**

Index of the current page of the PDF file. Page index starts from zero for the first page. The function **GetCurPage** makes available to get current page's index (this function can only be used in VC).

## 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).

You can set or get the value of Rotate by calling function **SetRotate** or **GetRotate** (these function can only be used in VC).

## 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 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.

To set or get the value of Zoomlevel, you can call function **SetZoomlevel** or **GetZoomlevel** (these function can only be used in VC).

## 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”

“Typewriter”(\*)

“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” (\*)  
“Caret 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” (\*)  
“Image Tool”(\*)  
“FileAttachment Tool” (\*)  
“Attach a file” (\*)

And so on.

You can call [CountTools](#) to learn how many tools are available in current version of ActiveX, and then call [GetToolByIndex](#) to get the tool names. You can also call **GetCurrentTool** or **SetCurrentTool** to get or set current tool’s name (these function can only be used in VC).

## Printer

### Type:

[IPDFPrinter](#), read-only

### Description:

Printer property returns an [IPDFPrinter](#) interface that you can use for managing the printer and sending the printout. In your application, you can use function **GetPrinter** to get this property (this function can only be used in VC).

## bHasFormFields

**Type:**

BOOL , read-only

**Description:**

If current document contains form fields, then bHasFormFields is True; otherwise, it is False. To get the value of bHasFormFields, you can use function **GetBHasFormFields** in your own application (this function can only be used in VC).

## DocumentInfo

**Type:**

[IPDFDocumentInfo](#)\*, read-only

**Description:**

Function **GetDocumentInfo** returns an [IPDFDocumentInfo](#) interface which you can use to retrieve document information such as Author, Creator, Creation Date, Keywords, ModDate, Producer Subject and Title (this function can only be used in VC).

## bHighlightFormFields

**Type:**

BOOL , read and write

**Description:**

Setting bHighlightFormFields to True will highlight all interactive form fields thereby having a better visual effects. By calling function **SetBHighlightFormFields** or **GetBHighlightFormFields**, you can set or get the value which indicates whether to highlight all interactive form fields or not (these function can only be used in VC).

## FormFieldsHighlightAlpha

**Type:**

short, read and write

**Description:**

Represent 256 levels of transparency of form field highlight color.

0=transparent; 255=opaque.

You can set or get access to the value of highlight alpha of form fields by using function **SetFormFieldsHighlightAlpha** or **GetFormFieldHighlightAlpha** (these function can only be used in VC).

## FormFieldsHighlightColor

**Type:**

OLE\_COLOR, read and write

**Description:**

Represent highlight color of form fields. You can call **SetFormFieldsHighlightColor**

or **GetFormFieldsHighlightColor** function to set or get the highlight color of form fields (these function can only be used in VC).

## ActiveXVersion

### Type:

BSTR, read-only

### Description:

Get the Version of ActiveX control. You can get the version of current registered ActiveX by calling **GetActiveXVersion** function (this function can only be used in VC).

**Note: In VC, you have to use these methods mentioned above to realize setting or getting corresponding properties; in other languages, you can directly get access to these properties. For example:**

code in VC:

```
FoxitReaderSDK.GetFilePath();
```

```
// the above statement returns the file path of current opened PDF file. But you can directly  
// use FilePath to get current opened PDF file path in other languages.
```

Code in VB, javascript, C# etc:

```
FoxitReaderSDK.FilePath
```

## 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.

## CloseFile

Close the currently loaded PDF file.

**Prototype:**

Void CloseFile()

**Parameter:**

[None]

**Return value:**

[None]

## SetFileStreamOption

Sets 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 result in 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]

## ShowBookmark

Show or hide the bookmark (outline) panel. Bookmark and outline refer to the same concept and they are used interchangeably in this document.

**Prototype:**

void ShowBookmark(BOOL bShow)

**Parameters:**

bShow - If this parameter is FALSE, the bookmark panel will be invisible.  
If this parameter is TRUE, the bookmark panel 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]

## 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]

## ShowCoverPage

Show the cover page of the PDF document during facing. Used in browser facing mode.

**Prototype:**

```
void ShowCoverPage(BOOL bShowCover);
```

**Parameters:**

**bShowCover** - A BOOL value indicating whether to show the cover page.

**Return value:**

[None]

**Note :**

This function will be removed in our next SDK version and it will be replaced with new function [SetFacingCoverLeft](#).

## 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]

### 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:**

```
Void SetViewRect(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.

## ConvertClientCoodToPageCood

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

**Prototype:**

```
BOOL ConvertClientCoodToPageCood (long nClientX, long nClientY,  
long* pnPageNum, 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
- pnPageNum - 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.

**Note:**

This function will be removed in next SDK version and it will be replaced with function [ConvertClientCoordToPageCoord](#)

## ConvertClientCoordToPageCoord

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

**Prototype:**

BOOL ConvertClientCoordToPageCoord (long nClientX, long nClientY, long\* pnPageNum, 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
- pnPageNum - 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.

## ConvertPageCoordToClientCoord

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

**Prototype:**

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

**Parameters:**

- nPageNum - 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

- pnClientY - outside the ActiveX control window's client area.  
- 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.

## GotoPageView2

Go to a specified position in a PDF document.

**Prototype:**

Void GotoPageView2 (ILink\_Dest \* link\_dest);

**Parameters:**

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

**Return value:**

[None]

**Note:**

This function will be named as [GoToPageDest](#) in future.

## 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 OnHypeLink .

**Return value:**

[None]

## GoToPage

Go to the specified page in a PDF document.

**Prototype:**

Void GoToPage (long page\_index);

**Parameters:**

page\_index - The specified index of the page you want to view.

**Return value:**

[None]

**Note:**

This function has taken the place of the former function GotoPage.

## GoToPage2

Go to a specified position in a PDF document.

**Prototype:**

Void GoToPage2 (long nPage, float PageX, float PageY);

**Parameters:**

- nPage - 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 nPage.(in PDF co-ordinate system)
- PageY - Specify the y coordinate of the position inside the PDF page which index is specified by nPage.

**Return value:**

[None]

**Note:**

This function will be changed to new function [GoToPagePos](#) in further version.

## GoToPagePos

Go to a specified position in a PDF document.

**Prototype:**

Void GoToPagePos (long nPage, float PageX, float PageY);

**Parameters:**

- nPage - 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 nPage.(in PDF co-ordinate system)
- PageY - Specify the y coordinate of the position inside the PDF page which index is specified by nPage.

**Return value:**

[None]

## GetVisibleLeftTopPage

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

**Prototype:**

Long GetVisibleLeftTopPage ();

**Parameter:**

[None]

**Return value:**

Long - the page index.

## Scroll

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

**Prototype:**

Void Scroll (long dx, long dy);

**Parameters:**

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

**Return value:**

[None]

**Note:**

This function will be replaced with new function [ScrollView](#) in further version.

## ScrollView

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

**rototype:**

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:**

long          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:**

The index of current page

## GoToNextPage

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

**Prototype:**

Void          GoToNextPage ();

**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.

### 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 notice 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 GotoPageView 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.

## 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.

## GotoPageView

Display and highlight the search result.

**Prototype:**

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

**Parameters:**

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

**Return value:**

[None]

**Note:**

This function will be removed and replaced with new function [GoToSearchResult](#) in future.

## 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 does have child item, 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, Stream objects 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 (\*)

Submits the form data to a specified URL

**Prototype:**

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

**Parameters:**

csDestination - The URL to submit to.

**Return Value:**

Return value indicates whether the operation is successful.

## 9) Drawing

### SetPencilColor (\*)

Set the color of this pen.

**Prototype:**

```
void SetPencilColor(OLE_COLOR penColor);
```

**Parameters:**

penColor - Contains an RGB color for the pen.

**Return Value:**

[None]

## SetPencilOpacity (\*)

Set the transparency of a pencil.

### Prototype:

void SetPencilOpacity(short nOpacity)

### Parameters:

nOpacity - A number from 0 to 255, identifying the alpha value.

### Return Value:

[None]

## SetPencilSize (\*)

Set the width of the pencil.

### Prototype:

void SetPencilSize(short nPenWidth);

### Parameters:

nPenWidth - Specifies the width of the pencil.

### Return Value:

[None].

## 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 nPage, float left, float bottom, float width, float height, BSTR BmpFileName, short alpha, short rotate);

### Parameters:

- nPage - The page number of the document to be Added image.
- left - The horizontal X position in which the watermark is placed. Starting from 0 at the left-most pixel.
- bottom - The horizontal y position in which the watermark 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]

### RunJScript (\*)

Use JavaScript to control some features.

### Prototype:

BSTR                      RunJScript (BSTR csJS)

**Return Value:**

The return value of the Javascript. It can be something like “0”, “hello”, “The Javascript runs successfully.”

## 11) Others

### GetSelectedText

Get currently selected text.

**Prototype:**

BSTR                      GetSelectedText ();

**Parameters:**

[None]

**Return value:**

The text that has been selected.

### 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.

### 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 top left corner of the rectangular region

top                        - Y-coordinate of the top left corner of the rectangular region

right                     - X-coordinate of the bottom right corner of the rectangular region

bottom                    - Y-coordinate of the bottom right corner of the rectangular region

**Return value:**

[None]

## SetHighlightColor(\*)

Set the highlight color.

**Prototype:**

void SetHighlightColor(OLE\_COLOR nColor);

**Parameters:**

nColor - Contains an RGB color for the pen.

**Return value:**

[None]

## SetHighlightType (\*)

Set the highlight type.

**Prototype:**

Void SetHighlightType (short nType);

**Parameters:**

nType - The type of the highlight.

**Return value:**

[None]

## RemoveAllHighlight (\*)

Remove all highlight in current opened 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 nPage)

**Parameters:**

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

**Return value:**

The text that has been extracted.

## 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 <= nIndex < 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 indicating 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]

## SetCurrentLanguage

The user interface of ActiveX can be switched to one of the 30+ languages dynamically. This requires an 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 an 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 name of the language file. Such as "lang\_en\_us.xml".

**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.

## 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 which 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 specify the pages have been exported.

**GetBitmap(\*)**

Render contents in a page to 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 Software Company.

**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 customers, you should call this function before calling any other ActiveX functions.

**Events:****BeforeDraw**

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

**Prototype:**

BeforeDraw (long dc)

**Parameters:**

dc - Handle to a device context.

**AfterDraw**

Triggered After the painting of the viewer contents is completed.

**Prototype:**

AfterDraw (long dc)

**Parameters:**

dc - Handle to a device context.

**OnZoomChange**

Triggered when you change the Zoomlevel property.

**Prototype:**

OnZoomChange ()

**Parameters:**

[None]

**OnPageChange**

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

**Prototype:**

OnPageChange ()

**Parameters:**

[None]

**OnOpenPassword**

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

**Prototype:**

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.

**OnHypeLink (\*)**

Triggered when the user is click on a hypertext,

**Prototype:**

OnHypeLink(BSTR linktype, BSTR linkdata, Link\_Dest\* dest, BOOL\* cancel)

**Parameters:**

- Linktype - A string containing information about the type of hyperlink.  
linktype sting are:
  - GoTo** move 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** move to a different PDF file stored on the local disk, if the a 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** launch an external application, if the a 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 contain additional information separated by character.

**OnSearchProgress**

Triggered when you search document,

**Prototype:**

OnSearchProgress (long pageNumber, long pageCount)

**Parameters:**

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

**OnOpenDocument**

Triggered when you open a document.

**Prototype:**

OnOpenDocument (BSTR filepath)

**Parameters:**

- Filepath - Path to the PDF file.

## OnCloseDocument

Triggered when you close a document.

**Prototype:**

OnCloseDocument (BSTR filepath)

**Parameters:**

Filepath - Path to the PDF file.

## OnDocumentChange

Triggered when the PDF document content change.

**Prototype:**

OnDocumentChange ()

**Parameters:**

[None]

## CustomFileGetSize

Triggered when use OpenCustomFile method to open PDF document.

**Prototype:**

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:**

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.

## OnContextMenu

Triggered when the context menu is called.

**Prototype:**

OnContextMenu (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.

## OnClick

Triggered when the left button is clicked.

### Prototype:

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:

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.

## 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 page number to start printing from.

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

printerRangeTo

**Type:**

short

**Description:**

Specify the page number to start printing to.

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

numOfCopies

**Type:**

short

**Description:**

Specify the number of copies to be printed.

**methods:****PrintWithDialog**

Display Windows dialog for sending print-outs.

**Prototype:**

```
void          PrintWithDialog();
```

**Parameters:**

[None]

**Return value:**

[None]

**PrintQuiet**

Sends 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 read Windows SDK documentation.

**Prototype:**

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

**Parameters:**

paperSize - Available paper sizes.

**Return value:**

[None]

## PDFOutline

Gives details regarding the PDF outline object.

### 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

FindResult class if an occurrence is found.

methods:

### GetFindPageNum

Get the page number.

**Prototype:**

long                    GetFindPageNum();

**Parameters:**

[None]

**Return value:**

The page index.

### GetFindFileName

Get the find file name.

**Prototype:**

BSTR                    GetFindFileName();

**Parameters:**

[None]

**Return value:**

The file name.