Foxit Reader ActiveX SDK 2.0

Programming Guide

2008.02

Index

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 be used and has rich features built inside it. A programmer can simply drag and drop the component into his application and instantly add PDF displaying functions to the application. The ActiveX allows users to navigate, zoom, rotate, scroll and print out PDF documents.

Version 2.0 incorporates advanced PDF features. It supports annotation and allows users to fill out, import or export PDF forms. Version 2.0 exposes more functions and events, giving programmers flexible control over the component and more access to the PDF documents.

Foxit offers two different sub-versions of ActiveX 2.0. One is standard version which doesn't include the following features: creating/editing annotation, importing/exporting form data, running javascript, converting PDF to text, etc. The other one is professional version which includes these features. You may choose the right version based on the requirement of your application. For standard version, you may simply purchase it online. For professional version, you need to contact 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 a star sign *.

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, to be 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 .

This is a SDK ActiveX, not a browser ActiveX. It will be used by programmers, not directly by end users. Foxit Software will develop a separate browser ActiveX and announce it once it becomes available.

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.

TUTORIAL 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”,””);

TUTORIAL 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

TUTORIAL 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.

TUTORIAL 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 function.

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

FoxitReaderSDK. SetRotate(1);

TUTORIAL 5: Print a PDF document

What you need to do is to 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 IPDFPrinter interface.

TUTORIAL 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.

TUTORIAL 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 IPDFOutline Interface.

TUTORIAL 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.

TUTORIAL 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 “Line Tool”.

Reference

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 BSTR, read-only

Full path to the current opened PDF file. If no PDF file is opened, the

property is an empty string.

PageCount long, read-only

Total number of pages in the current opened PDF file.

CurPage long, read-only

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

Rotate short, read and write

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 long, read and write

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.

CurrentTool BSTR, read and write

Read and set the current tool. The value can be one of following strings:
“Hand Tool”
“ZoomOut Tool”
“ZoomIn Tool”
“TypeWriter Tool” **(*)**
“Select Text Tool”
“Find Text Tool”
“Snapshot Tool”
“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.

Printer IPDFPrinter, read-only

Printer property returns an IPDFPrinter interface which you can use for managing the printer and sending the printout.

bHasFormFields (*) BOOL , read-only

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

DocumentInfo IPDFDocumentInfo*, read-only

Returns an IPDFDocumentInfo interface which you can use to retrieve document information such as Author, Creator, Creation Date, Keywords, ModDate, Producer Subject and Title

bHighlightFormFields (*) BOOL , read and write

Setting bHighlightFormFields to True will highlight all interactive form fields thereby create better visual effect.

FormFieldsHighlightAlpha (*) short, read and write

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

FormFieldsHighlightColor (*) OLE_COLOR, read and write

Represent highlight color of form fields.

Methods:

1) Open and close PDF File

OpenFile

Open a PDF file from a local disk or from a http server.
Note : The file will not be locked if it is opened by this method., it can be opened by other program.

Prototype:
BOOL OpenFile (BSTR FilePath, long Password)

Parameters

FilePath - Path to the PDF file, including file extension.

Or a URL to a HTTP server,

Password - A password string.

If no password is needed, you may specify empty string.

Return value:

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

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 - A password string.

If no password is needed, you may specify empty string.

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 - A password string.

If no password is needed, you may specify 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 programcall 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 descripton of CustomFileGetSize and CustomFileGetBlock for more details.

Prototype:
BOOL OpenCustomFile(BSTR Password)

Parameter:

Password - A password string. If no password is needed, simply use 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]

2) 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 SearchString, 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 FileName , BSTR SearchString, BOOL bMatchCase, BOOL bMatchWholeWord)

Parameters:

FileName - Full name of the PDF file, including full path.
SearchString - 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]

SearchAndHighlightAllTextOnPage

Highlight all instance of a given keyword in a specified page,

Prototype:

void SearchAndHighlightAllTextOnPage(BSTR searchstring, boolean bMatchCase, boolean bMatchWholeWord, long PageNo);

Parameters:

PageNo - Number of the page you want to search.

Return Value:

[None]

3) 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 (if any) will be returned

Return value:

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

4) Layout

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]

5) 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:

A IStream interface supports reading and writing data to stream objects , Stream objects contain the PDF file data .

6) Others

ExistForwardStack

Detect the existence of next view.

Quite often, when a user navigates through a PDF file, he would like to go back to 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.

Prototype:

BOOL ExistForwardStack ();

Parameters : [None]

Return value:

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

GoForwardStack

Go to 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

Go To previous view,

Prototype:

void GoBackwardStack ();

Parameters: [None]

*Return value: [*None]

ShowTitleBar

Show or hide the title bar;

Prototype:
void ShowTitleBar(BOOL Show);

Parameters:

Show - 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 Show);

Parameters:

Show - 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, short show)

Parameters
nIndex - The index of the button.

Show - If this parameter is 0, the toolbar button will be invisible. If this parameter is nonzero, 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 Show)

Parameters:

Show - 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 Show);

Parameters:

Show - 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]

GetSelectedText

Get currently selected text.

Prototype:

BSTR GetSelectedText();

Parameters: [None]

Return value: The text that has been selected.

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]

SetViewRect

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.

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]

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]

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]

RemoveAllHighlight

Remove all highlight in current opened document.

Prototype:

void RemoveAllHighlight()

Parameters:

[None]

Return value:

[None]

ConvertClientCoodToPageCood

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

Prototype:

Boolean 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
pdPageX - for returning x coordinate of the point inside the PDF page (in PDF co-ordinate system)
pdPageY - 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 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.

FindFormFieldsTextFirst (*)

Search the text in form fields.

Prototype:

Boolean FindFormFieldsTextFirst(BSTR searchstring, boolean 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:

boolean FindFormFieldsTextNext()

Parameters:

[None]

Return value:

Return value indicates whether the searched string is found.

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()

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.

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.”.

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 .

ShowDocJsDialog (*)

Popup Document Javascript Dialog.

ShowJsConsoleDialog (*)

Popup Javascript Console Dialog.

ExportFormToFDFFile (*)

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

Prototype:

boolean ExportFormToFDFFile(BSTR FDFFileName)

Parameters:

FDFFileName - 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:

boolean ImportFormFromFDFFile(BSTR FDFFileName)

Parameters:

FDFFileName - fdf file path.

Return Value:

Return value indicates whether the operation is successful.

ExportAnnotsToFDFFile (*)

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

Prototype:

boolean ExportAnnotsToFDFFile(BSTR FDFFileName)

Parameters:

FDFFileName - 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:

boolean ImportAnnotsFromFDFFile(BSTR FDFFileName)

Parameters:

FDFFileName - fdf file path.

Return Value:

Return value indicates whether the operation is successful.

SubmitForm (*)

Submits the form data to a specified URL

Prototype:

boolean SubmitForm(BSTR csDestination)

Parameters:

csDestination - The URL to submit to.

Return Value:

Return value indicates whether the operation is successful.

SetCurrentLanguage

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

Prototype:

void SetCurrentLanguage(short LanguageID);

Parameters:

LanguageID - Language identifier. A value from 0 to 30 to represent different languages.

Return value:

[None]

UnLockActiveX

Unlock the Activex using license key received from Foxit Software Company.

Prototype:

void UnLockActiveX(BSTR lisence_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]

Remarks:

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, boolean* cancel)

Parameters:

Password - A password string.

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, boolean* 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 1, otherwise, followed 0.

URI open a 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 .

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.

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.

IPDFPrinter interface

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

IPDFPrinter Properties:

BSTR printerName;

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

PrinterRangeMode printerRangeMode;

- 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,

short printerRangeFrom;

- You must first set the PrinterRangeMode to PRINT_RANGE_SELECTED

short printerRangeTo;

- You must first set the PrinterRangeMode to PRINT_RANGE_SELECTED

short numOfCopies;

- Number of copies for printing.

IPDFPrinter methods:

PrintWithDialog();

- Display Windows dialog for sending print-outs.

PrintQuiet();

- Send the printout.

SetPaperSize(long paperSize);

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

IPDFOutline interface

IPDFOutline methods:

Void NavigateOutline()

Follows the destination specified from the outline object.

BSTR GetOutlineTitle()

Get the title of the outline Object.

IPDFDocumentInfo interface

With IPDFDocumentInfo interface you can get the pdf document information.

IPDFDocumentInfo Properties:

BSTR Author

BSTR Subject

BSTR CreatedDate

BSTR ModifiedDate

BSTR Keywords

BSTR Creator

BSTR Producer

BSTR Title