



# API Reference for Application Communication



Fast, Affordable & Secure PDF Solutions  
Foxit Software Inc.

© Foxit Software Incorporated. 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.

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

Preface .....	5
What's in this guide? .....	5
Who should read this guide? .....	5
Related documentation .....	5
OLE Automation .....	6
FoxitExch.App .....	6
Exit .....	7
GetActiveDoc .....	7
GetAvDoc .....	7
GetNumAvDoc .....	8
Show .....	8
Hide .....	8
FoxitExch.AVDoc .....	9
Close .....	9
GetAVPageView .....	10
GetPDDoc .....	10
IsValid .....	10
Open .....	11
OpenWithPassword .....	11
OpenDiskFileAsMemPDF .....	12
CreateBlankDoc .....	12
FoxitExch.AVPageView .....	13
GetPageNum .....	13
Goto .....	13
FoxitExch.PDBookmark .....	14
GetByTitle .....	14
IsValid .....	15
Perform .....	15
FoxitExch.PDDoc .....	15
Close .....	16
Create .....	17
GetInfo .....	17
GetJSObject .....	17
GetNumPages .....	18
InsertPages .....	18
Open .....	19
OpenAVDoc .....	19
Save .....	20
SetInfo .....	21

RotatePage .....	21
GetFieldValue .....	22
SetFieldValue .....	22
AddWatermark .....	23
OCRAndExportToExcel .....	24
CreateWatermarkElementInfo .....	25
FoxitWatermarkElementInfo .....	25
Serialize .....	27
Foxit PhantomPDF.Creator .....	27
CombineFiles .....	27
Foxit PhantomPDF.CombineFlags .....	28
COMBINE_DEFAULT .....	28
COMBINE_STOP_CONVERTFAIL .....	28
COMBINE_ADD_CONTENTS .....	28
COMBINE_TRAVEL_SUBDIR .....	29
Foxit PhantomPDF.PDSaveFlags .....	29
PDSaveIncremental .....	29
PDSaveFull .....	29
PDSaveCopy .....	29
PDSaveLinearized .....	29
PDSaveCollectGarbage .....	29
<b>FoxitPDFCreatorForOffice .....</b>	<b>30</b>
FoxitPDFCreatorForOffice.PDFCreator .....	30
ConvertToPDFEx .....	30
GetCurrentConversionSettings .....	30
FoxitPDFCreatorForOffice. ISettings .....	31
<b>Demo for VBA .....</b>	<b>32</b>
A simple VBA code skeleton of your application .....	32
Adding a text watermark to the center of all pages .....	32
Adding an image watermark to the center of all pages .....	33
Getting/Setting a PDF form's value .....	34
OCR and export the document to an Excel document .....	37
Combining several documents to a PDF .....	37
Converting non-PDF to PDF .....	38

# Preface

The Foxit® PhantomPDF® Software Development Kit (SDK) provides a set of API calls for creating PDFs from other document types and controlling PhantomPDF to perform other tasks without manual intervention.

## What's in this guide?

This document provides a detailed reference of all the APIs that are used to communicate with PhantomPDF (including the included Microsoft Office add-ins).

## Who should read this guide?

This guide is for developers that need to communicate with PhantomPDF from other applications, or who are writing tools that will use PhantomPDF to automatically operate on PDF documents.

You can use all the APIs easily if you are familiar with VBA or C++.

## Related documentation

<b>For information about</b>	<b>See</b>
Foxit PhantomPDF_Manual.pdf	<i>Foxit PhantomPDF User Manual10.0.pdf</i>

# OLE Automation

This chapter describes the objects, data types, and methods in the OLE automation interface. FoxitPhantomPDF supports dual interfaces, so the methods all have a return type of HRESULT. The following table summarizes the available objects and data types.

<b>Object</b>	<b>Description</b>
<a href="#">FoxitExch.App</a>	The application itself.
<a href="#">FoxitExch.AVDoc</a>	A document as seen in the user interface.
<a href="#">FoxitExch.AVPageView</a>	The area of the window that displays the contents of a page.
<a href="#">FoxitExch.PDBookmark</a>	A bookmark in a PDF file.
<a href="#">FoxitExch.PDDoc</a>	The underlying PDF representation of a document.
<a href="#">FoxitWatermarkElementInfo</a>	Represents a watermark structure.
<a href="#">Foxit PhantomPDF.Creator</a>	A creator object that is used to convert a non-PDF file to PDF.
<a href="#">Foxit PhantomPDF.CombineFlags</a>	An enum used for the “Creator.CombineFiles” parameter uCombineFlags.
<a href="#">Foxit PhantomPDF.PDSaveFlags</a>	An enum used for the “PDDoc.Save’s” parameter nType.

## FoxitExch.App

The Foxit PhantomPDF application itself. This is a creatable interface. From the application layer, you can control the appearance of Foxit PhantomPDF and whether the Foxit PhantomPDF application window appears.

### Methods

The App object has the following methods.

<b>Method</b>	<b>Description</b>
<a href="#">Exit</a>	Exits Foxit PhantomPDF.
<a href="#">GetActiveDoc</a>	Gets the frontmost document.
<a href="#">GetAVDoc</a>	Gets a FoxitExch.AVDoc object via its index within the list of open AVDoc objects.
<a href="#">GetNumAVDocs</a>	Gets the number of open FoxitExch.AVDoc objects.
<a href="#">Hide</a>	Hides the Foxit PhantomPDF application.

---

[Show](#)

Shows the PhantomPDF application.

---

## Exit

Exits Foxit PhantomPDF. Applications should call App.Exit before exiting.

**Note:** Close all the documents before calling this method.

## Syntax

```
VARIANT_BOOL Exit();
```

## Returns

Returns -1 if the entire shutdown process succeeded. This includes closing any open documents, releasing OLE references, and finally exiting the application. If any step fails, the function returns 0, and the application continues running. This method does not work if the application is visible (if the user is in control of the application). In such cases, if the Show method had previously been called, you can call Hide and then Exit.

## GetActiveDoc

Gets the frontmost document.

## Syntax

```
LPCDISPATCH GetActiveDoc();
```

## Returns

The LPCDISPATCH for the frontmost FoxitExch.AVDoc object. If there are no documents open, it returns NULL.

## See Also

App.[GetAVDoc](#)

## GetAvDoc

Gets a FoxitExch.AVDoc object from its index within the list of open AVDoc objects. Use

App.[GetNumAVDocs](#) to determine the number of FoxitExch.AVDoc objects.

## Syntax

```
LPCDISPATCH GetAVDoc (long nIndex);
```

## Parameters

---

nIndex	The index of the document to get.
--------	-----------------------------------

---

## Returns

The LPDISPATCH for the specified FoxitExch.AVDoc document, or NULL if nIndex is greater than the number of open documents

## GetNumAvDoc

Gets the number of open FoxitExch.AVDoc objects.

### Syntax

```
long GetNumAVDocs();
```

### Returns

The number of open FoxitExch.AVDoc objects.

### See Also

App.[GetActiveDoc](#)

App.[GetAVDoc](#)

## Show

Shows the Foxit PhantomPDF application. When the viewer is shown, the user is in control, and the Foxit PhantomPDF application does not automatically exit when the last automation object is destroyed. However, it will exit if no documents are being displayed.

### Syntax

```
VARIANT_BOOL Show();
```

### Returns

-1 if successful, 0 if not.

### See Also

App.[Hide](#)

## Hide

Hides the Foxit PhantomPDF application. When the viewer is hidden, the user has no control over it, and the Foxit PhantomPDF application exits when the last automation object is closed

### Syntax

```
VARIANT_BOOL Hide();
```

### Returns

-1 if successful, 0 if not.



## See Also

[App.Show](#)

# FoxitExch.AVDoc

A view of a PDF document in a window. This is a creatable interface. There is one AVDoc object per displayed document. Unlike a PDDoc object, an AVDoc object has a window associated with it.

## Methods

The AVDoc object has the following methods.

Method	Description
<a href="#">Close</a>	Closes a document.
<a href="#">GetAVPageView</a>	Gets the FoxitExch.AVPageView associated with a FoxitExch.AVDoc.
<a href="#">GetPDDoc</a>	Gets the FoxitExch.PDDoc associated with a FoxitExch.AVDoc.
<a href="#">Open</a>	Opens a file.
<a href="#">IsValid</a>	Determines whether the FoxitExch.AVDocis still valid.
<a href="#">OpenWithPassword</a>	Opens a file with a password.
<a href="#">OpenDiskFileAsMemPDF</a>	Opens a file as an in-memory document.
<a href="#">CreateBlankDoc</a>	Creates a blank document.

## Close

Closes a document.

## Syntax

```
VARIANT_BOOL Close (long bNoSave);
```

## Parameters

bNoSave	If a positive number, the document is closed without saving it. If 0 and the document has been modified, the user is asked whether or not the file should be saved.
---------	---

## Returns

Always returns -1, even if no document is open.

## See Also

[AVDoc.Open](#)

[AVDoc.OpenWithPassword](#)

[AVDoc.OpenDiskFileAsMemPDF](#)

[PDDoc.Open](#)

[PDDoc.OpenAVDoc](#)

## GetAVPageView

Gets the FoxitExch.AVPageView associated with a FoxitExch.AVDoc.

### Syntax

```
LPDISPATCH GetAVPageView ();
```

### Returns

The LPDISPATCH for the FoxitExch.AVPageView or NULL if no document is open.

### See Also

[AVDoc.GetPDDoc](#)

## GetPDDoc

Gets the FoxitExch.PDDoc associated with a FoxitExch.AVDoc.

### Syntax

```
LPDISPATCH GetPDDoc ();
```

### Returns

The LPDISPATCH for the FoxitExch.PDDoc or NULL if no document is open.

### See Also

[AVDoc.GetAVPageView](#)

## IsValid

Determines whether the FoxitExch.AVDoc is still valid. This method only checks if the document has been closed or deleted; it does not check the internal structure of the document.

### Syntax

```
VARIANT_BOOL IsValid ();
```

### Returns

-1 if the document can still be used, 0 otherwise.

### See Also

App.[GetAVDoc](#)

## Open

Opens a file. A new instance of FoxitExch.AVDoc must be created for each displayed PDF file.

### Syntax

```
VARIANT_BOOL Open (BSTR szFullPath, BSTR szTempTitle);
```

### Parameters

---

szFullPath	The full path of the file to open.
szTempTitle	An optional title for the window in which the file is opened. If szTempTitle is NULL or the empty string, it is ignored. Otherwise, szTempTitle is used as the window title.

---

### Returns

-1 if the file was opened successfully, 0 otherwise.

### See Also

AVDoc.[Close](#)

AVDoc.[OpenWithPassword](#)

AVDoc.[OpenDiskFileAsMemPDF](#)

PDDoc.[Close](#)

PDDoc.[Open](#)

PDDoc.[OpenAVDoc](#)

## OpenWithPassword

Opens a file with a password. A new instance of FoxitExch.AVDoc must be created for each displayed PDF file

### Syntax

```
VARIANT_BOOL OpenWithPassword (BSTR szFullPath, BSTR szpassword);
```

### Parameters

---

szFullPath	The full path of the file to open.
szpassword	The password.

---

### Returns

-1 if the file was opened successfully, 0 otherwise.

## See Also

[AVDoc.Close](#)

[AVDoc.Open](#)

[AVDoc.OpenDiskFileAsMemPDF](#)

[PDDoc.Close](#)

[PDDoc.Open](#)

[PDDoc.OpenAVDoc](#)

## OpenDiskFileAsMemPDF

Opens a file as an in-memory document. A new instance of FoxitExch.AVDoc must be created for each displayed PDF file

### Syntax

```
VARIANT_BOOL OpenDiskFileAsMemPDF (BSTR szFullPath, BSTR szTempTitle);
```

### Parameters

---

szFullPath	The full path of the file to open.
szTempTitle	An optional title for the window in which the file is opened. If szTempTitle is NULL or the empty string, it is ignored. Otherwise, szTempTitle is used as the window title.

---

### Returns

-1 if the file was opened successfully, 0 otherwise.

## See Also

[AVDoc.Close](#)

[AVDoc.Open](#)

[AVDoc.OpenWithPassword](#)

[PDDoc.Close](#)

[PDDoc.Open](#)

[PDDoc.OpenAVDoc](#)

## CreateBlankDoc

Creates a blank document. A new instance of FoxitExch.AVDoc must be created for each displayed PDF file.

## Syntax

```
VARIANT_BOOL CreateBlankDoc(FLOAT fWidth, FLOAT fHight);
```

## Parameters

fWidth	The page width.
fHight	The page height.

## Returns

-1 if the file was opened successfully, 0 otherwise.

# FoxitExch.AVPageView

The area of the Foxit PhantomPDF application's window that displays the contents of a document's page. This is a non-creatable interface. Every AVDoc object has an AVPageView object and vice versa. The object provides access to the PDDoc and PDPage objects for the document being displayed.

## Methods

The AVPageView object has the following methods.

Method	Description
<a href="#">GetPageNum</a>	Gets the page number of the current page.
<a href="#">Goto</a>	Goes to the specified page.

## GetPageNum

Gets the page number of the current page. Pages are ordered numerically starting with 0.

## Syntax

```
long GetPageNum();
```

## Returns

The current page's page number.

## See Also

[PDDoc.GetNumPages](#)

## Goto

Goes to the specified page.

## Syntax

```
VARIANT_BOOL GoTo(long nPage);
```

## Parameters

---

nPage	Page number of the destination page. The first page in a PDDoc object is page 0.
-------	--

---

## Returns

-1 if the application successfully went to the page, 0 otherwise.

# FoxtExch.PDBookmark

A bookmark for a page in a PDF file. This is a creatable interface. Each bookmark has a title that appears on screen, and an action that specifies what happens when a user clicks on the bookmark.

Bookmarks can either be created interactively by the user through the Foxit PhantomPDF application's user interface or programmatically generated. The typical action for a user-created bookmark is to move to another location in the current document, although any action can be specified. It is not possible to create a bookmark with OLE—only to destroy one.

## Methods

The PDBookmark object has the following methods.

---

Method	Description
<a href="#">GetByTitle</a>	Gets the bookmark that has the specified title.
<a href="#">IsValid</a>	Determines whether the bookmark is valid.
<a href="#">Perform</a>	Performs a bookmark's Mouse Up action(s).

---

## GetByTitle

Gets the bookmark that has the specified title. The FoxtExch.PDBookmark object is set to the specified bookmark as a side effect of the method; it is not the method's return value. You cannot enumerate bookmark titles with this method.

## Syntax

```
VARIANT_BOOL GetByTitle(LPDISPATCH iFoxitPDDoc, BSTR bookmarkTitle);
```

## Parameters

---

iFoxitPDDoc	The LPDISPATCH for the document (FoxtExch.PDDocobject) containing the bookmark.
-------------	---

---

---

bookmarkTitle	The title of the bookmark to get. The capitalization of the title must match that in the bookmark.
---------------	--

---

## Returns

-1 if the specified bookmark exists and is valid (the method determines this using the `PDBookmark.IsValid` method), 0 otherwise.

## IsValid

Determines whether the bookmark is valid. This method only checks whether the bookmark has been deleted; it does not thoroughly check the bookmark's data structures.

## Syntax

```
VARIANT_BOOL IsValid();
```

## Returns

-1 if the bookmark is valid, 0 otherwise.

## Perform

Performs a bookmark's action.

## Syntax

```
VARIANT_BOOL Perform(LPDISPATCH iFoxitAVDoc);
```

## Parameters

---

iFoxitAVDoc	The LPDISPATCH for the FoxitExch.AVDoc in which the bookmark is located.
-------------	--

---

## See Also

`PDBookmark.IsValid`

# FoxitExch.PDDoc

The underlying PDF representation of a document. This is a creatable interface. The PDDoc object is the hidden object behind every AVDoc object.

Through PDDoc objects, your application can perform most of the Organize menu items from Foxit PhantomPDF (insert pages, rotate pages, and so on), and set and retrieve document metadata fields.

## Methods

The PDDoc object has the following methods.

<b>Method</b>	<b>Description</b>
<a href="#">Close</a>	Closes a file.
<a href="#">Create</a>	Creates a new FoxitExch.PDDoc.
<a href="#">GetInfo</a>	Gets the value of a specified key in the document's Info dictionary.
<a href="#">GetJSObject</a>	Gets a dual interface to the JavaScript object associated with the PDDoc.
<a href="#">GetNumPages</a>	Gets the number of pages in a file.
<a href="#">InsertPages</a>	Inserts the specified pages from the source document after the indicated page within the current document.
<a href="#">Open</a>	Opens a file.
<a href="#">OpenAVDoc</a>	Opens a window and displays the document in it.
<a href="#">Save</a>	Saves a document.
<a href="#">SetInfo</a>	Sets the value of a key in a document's Infodictionary.
<a href="#">RotatePage</a>	Rotates the specified pages in the source document.
<a href="#">GetFieldValue</a>	Gets the specified field value.
<a href="#">SetFieldValue</a>	Sets the specified field value.
<a href="#">AddWatermark</a>	Adds a watermark.
<a href="#">OCRAndExportToExcel</a>	OCRs and exports the document to an Excel file.
<a href="#">CreateWatermarkElementInfo</a>	Creates a WatermarkElementInfo object.

## Close

Closes a file.

### Syntax

```
VARIANT_BOOL Close();
```

### Returns

-1 if the document was closed successfully, 0 otherwise.

### See Also

[AVDoc.Close](#)

[AVDoc.Open](#)

[AVDoc.OpenWithPassword](#)

[AVDoc.OpenDiskFileAsMemPDF](#)



[PDDoc.Open](#)

[PDDoc.OpenAVDoc](#)

## Create

Creates a new FoxitExch.PDDoc.

### Syntax

```
VARIANT_BOOL Create();
```

### Returns

-1 if the document is created successfully, 0 if it is not or if the Foxit PhantomPDF application does not support editing.

## GetInfo

Gets the value of a specified key in the document's Info dictionary.

### Syntax

```
BSTR GetInfo(BSTR szInfoKey);
```

### Parameters

szInfoKey	The key whose value is obtained.
-----------	----------------------------------

### Returns

The string if the value was read successfully. Returns an empty string if the key does not exist or its value cannot be read.

### See Also

[PDDoc.SetInfo](#)

## GetJSObject

Gets a dual interface to the JavaScript object associated with the PDDoc. This allows automation clients full access to both built-in and user-defined JavaScript methods available in the document.

### Syntax

```
IDispatch* GetJSObject();
```

### Returns

The interface to the JavaScript object if the call succeeded, NULL otherwise.

**Note:**

Currently the supported JavaScript Properties include:

Property	Description
<a href="#">console</a>	The IJSconsole object.
<a href="#">BookMarkRoot</a>	The Bookmark root. IJSBookmark object. Only supports the createChild method, and the Name and Children properties.

## GetNumPages

Gets the number of pages in a file.

### Syntax

```
long GetNumPages();
```

### Returns

The number of pages, or -1 if the number of pages cannot be determined.

### See Also

AVPageView.[GetPageNum](#)

## InsertPages

Inserts the specified pages from the source document after the indicated page within the current document.

### Syntax

```
VARIANT_BOOL InsertPages(long nInsertPageAfter,LPDISPATCH iPDDocSource,long nStartPage, long nNumPages, long bBookmarks);
```

### Parameters

nInsertPageAfter	The page in the current document after which pages from the source document are inserted. The first page in a PDDoc object is page 0. If -1, insert the page before the beginning of the file; if -2, insert the page after the end of the file.
iPDDocSource	The LPDISPATCH for the FoxitExch.PDDoc containing the pages to insert.
nStartPage	The first page in iPDDocSource to be inserted into the current document.
nNumPages	The number of pages to be inserted.
bBookmarks	If a positive number, bookmarks are copied from the source document. If 0, they are not.

## Returns

-1 if the pages were successfully inserted. Returns 0 if they were not or if the Foxit PhantomPDF application does not support editing.

## See Also

PDDoc.[GetNumPages](#)

## Open

Opens a file. A new instance of FoxitExch.PDDoc must be created for each open PDF file.

## Syntax

```
VARIANT_BOOL Open(BSTR szFullPath);
```

## Parameters

---

szFullPath	Full path of the file to be opened.
------------	-------------------------------------

---

## Returns

-1 if the document was opened successfully, 0 otherwise.

## See Also

AVDoc.[Close](#)

AVDoc.[Open](#)

AVDoc.[OpenWithPassword](#)

AVDoc.[OpenDiskFileAsMemPDF](#)

PDDoc.[Close](#)

PDDoc.[OpenAVDoc](#)

## OpenAVDoc

Opens a window and displays the document in it.

## Syntax

```
LPDISPATCH OpenAVDoc(BSTR szTitle);
```

## Parameters

---

szTitle	The title to be used for the window. A default title is used if szTitle is NULL or an empty string.
---------	---

---

## Returns

The LPDISPATCH for the FoxitExch.AVDoc that was opened, or NULL if the open fails.

## See Also

[AVDoc.Close](#)

[AVDoc.Open](#)

[AVDoc.OpenWithPassword](#)

[AVDoc.OpenDiskFileAsMemPDF](#)

[PDDoc.Close](#)

[PDDoc.Open](#)

## Save

Saves a document.

## Syntax

```
VARIANT_BOOL Save(short nType, BSTR szFullPath);
```

## Parameters

---

nType	Specifies the way in which the file should be saved. nType is a logical OR of one or more of the following flags: PDSaveIncremental — Write changes only, not the complete file. This will always result in a larger file, even if objects have been deleted. PDSaveFull — Write the entire file to the filename specified by szFullPath. PDSaveCopy — Write a copy of the file into the file specified by szFullPath, but keep using the old file. This flag can only be specified if PDSaveFull is also used. PDSaveCollectGarbage — Remove unreferenced objects; this often reduces the file size, and its usage is encouraged. This flag can only be specified if PDSaveFull is also used. PDSaveLinearized — Save the file optimized for the web. This allows the PDF file to be byte-served. This flag can only be specified if PDSaveFull is also used.
-------	--

**Note:** If you save a file optimized for the web using the PDSaveLinearized flag, you must follow this sequence:

1. Open the PDF file with [PDDoc.Open](#).
  2. Call [PDDoc.Save](#) using the PDSaveLinearized flag.
  3. Call [PDDoc.Close](#).
-

---

	This allows batch optimization of files.
szFullPath	The new path to the file, if any.

---

## Returns

-1 if the document was successfully saved. Returns 0 if it was not or if the Foxit PhantomPDF application does not support editing.

## SetInfo

Sets the value of a key in a document's Info dictionary.

## Syntax

```
VARIANT_BOOL SetInfo(BSTR szInfoKey, BSTR szBuffer);
```

## Parameters

---

szInfoKey	The key whose value is set.
szBuffer	The value to be assigned to the key.

---

## Returns

-1 if the value was added successfully, 0 if it was not or if the Foxit PhantomPDF application does not support editing.

## See Also

PDDoc.[GetInfo](#)

## RotatePage

Rotates specified pages.

## Syntax

```
VARIANT_BOOL RotatePage(SHORT nPage, SHORT nRotate);
```

## Parameters

---

nPage	The pages index that begins with 0.
nRotate	nRotate , 0 for 0°, 1 for 90°, 2 for 180°, -1 for -90°

---

## Return value

-1 if successful, 0 if not.

## Remarks

The parameter nRotate is only 0, 1, 2 or -1, and any other value will result in failure.

## GetFieldValue

Gets the value of a specified form.

### Syntax

```
BSTR GetFieldValue(BSTR FieldName);
```

### Parameters

---

FieldName	The form name.
-----------	----------------

---

### Return value

Value of the specified form.

### Remarks

At present for the form types supported, see the following:

- ✧ Text Field
- ✧ Push Button
- ✧ List Box
- ✧ Combo Box
- ✧ Radio Button
- ✧ Check Box

## SetFieldValue

Sets the value of a specified form.

### Syntax

```
long SetFieldValue(BSTR FieldName, BSTR value);
```

### Parameters

---

FieldName	The form name.
value	Value to set.

---

### Return value

-1 Foxit PhantomPDF is invalid.

- 0 Successful.
- 1 The specified form does not exist.
- 2 Can't find the specified item's value.
- 3 Setting a value for the form field is not supported.
- 4 The document is invalid.

## Remarks

At present for the form type supported, see the following:

- ✧ Text Field
- ✧ Push Button
- ✧ List Box
- ✧ Combo Box
- ✧ Radio Button
- ✧ Check Box

## AddWatermark

Adds a watermark.

### Syntax

```
long AddWatermark(IWatermarkElementInfo* pWatermarkInfo);
```

### Parameters

---

pWatermarkInfo	A watermark information object.
----------------	---------------------------------

---

### Return value

- 0 Successful.
- 1 Invalid parameter.
- 2 The document has no permissions.
- 3 Foxit PhantomPDF has no permissions.
- 1 Foxit PhantomPDF was not started.
- 2 Inserting watermarks failed.

## Remarks

The parameter `pWatermarkInfo` is always created through calling `FoxitExch.PDDoc.CreateWatermarkElementInfo`.

## OCRAndExportToExcel

OCR and export the document to an Excel document.

### Syntax

```
long OCRAndExportToExcel(BSTR ExcelPathname, SHORT start, SHORT end, VARIANT_BOOL  
bAllPages,VARIANT_BOOL niText);
```

### Parameters

<code>ExcelPathname</code>	An Excel Pathname.
<code>start</code>	Page index of start page, ignore if <code>bAllPages</code> is true.
<code>end</code>	Page index of end page, ignore if <code>bAllPages</code> is true.
<code>bAllPages</code>	Whether to work on all pages.
<code>niText</code>	Text-based document or Image-based document mode. Text-based document is the default mode. In the Text-based document mode, <code>niText</code> is True: Images will not be OCR'd. In the Image-based document mode, <code>niText</code> is False: Images will be OCR'd.

### Return value

- 0 Successful.
- 1 Invalid parameter.
- 2 The document has no permissions.
- 3 Foxit PhantomPDF has no permission.
- 4 dest file no permission
- 1 app did not start
- 2 The document has been modified.
- 3 The conversion failed.

### Remarks

The parameter `ExcelPathname` must include `.xlsx` as a suffix, otherwise it will automatically be appended. The parameters `Start` and `End` both begin with 0, and will both be ignored if `bAllPages` is true.

If `ExcelPathname` already exists, the old file will be replaced with the newly generated file automatically.



## CreateWatermarkElementInfo

Creates a WatermarkElementInfo object.

### Syntax

```
IWatermarkElementInfo* CreateWatermarkElementInfo();
```

### Returns

A IWatermarkElementInfo object.

### See Also

PDDoc.[AddWatermark](#)

## FoxitWatermarkElementInfo

Represents information about a watermark. FoxitWatermarkElementInfo is usually a parameter of the interface Document.AddWatermark.

### Methods

The FoxitWatermarkElementInfo object has the following methods.

Method	Description
<a href="#">Serialize</a>	Serialized as a string.

### Properties

The FoxitWatermarkElementInfo object has the following Properties.

Property	Description
Type	0-Text, 1-File. The default value is 0.
WMFile	Watermark file path. Ignored if Type is 0.
WMText	Watermark text. Ignored if Type is 1.
FontName	Watermark font name. Ignored if Type is 1.
FontSize	Watermark font size. Ignored if Type is 1 or WMScale is greater than or equal to zero. ( 1-99999 )
TextColorRef	A color, for example 0x000000, 0xFFFFFFFF and 0x888888. Ignored if Type is 1.
Rotation	Rotation Angle (0 ~ 360) °.

Opacity	Opacity (0 ~ 1).
WMScale	Scale relative to the target page, (-1 ~ 1). -1 means no scale, and FontSize will be ignored if WMScale's value is set. Note: If Type is 0, the watermark is scaled relative to the target page; If Type is 1, the watermark is scaled relative to the original image file (i.e. "absolute scale").
Top	0 - Appear behind page, 1 - Appear on top of page.
VerticalDistance	Vertical offset, inches, (-100000.00 ~ 100000.00)
VerticalDistanceFrom	Top 0, Center 1, Bottom 2
HorizontalDistance	Horizontal offset, inches, (-100000.00 ~ 100000.00)
HorizontalDistanceFrom	Left 0, Center 1, Right 2
Start	The first page where the watermark will be placed, with the page index beginning with 0. Or, -1 to start at the beginning of the document.
End	The last page where the watermark will be placed, with the page index beginning with 0. Or, -1 to place watermarks until the end of the document.
Even	Whether to add watermarks to even-numbered pages only. 1 or True can be used to do this, otherwise 0 or False can be used.
Odd	Whether to add watermarks to odd-numbered pages only. 1 or True can be used to do this, otherwise 0 or False can be used.
WMFilePageIndex	The page index of the Watermark file. Beginning from 0. Ignore if Type is 0.

## Remarks

If both Start and End are set to -1, the document will be placed on all pages of the document.

Common colors for TextColorRef:

Constants	Value	Description
vbBlack	RGB(0, 0, 0)	Black
vbRed	RGB(255, 0, 0)	Red
vbGreen	RGB(0, 255, 0)	Green

vbYellow	RGB(255, 255, 0)	Yellow
vbBlue	RGB(0, 0, 255)	Blue
vbMagenta	RGB(255, 0, 255)	Magenta
vbCyan	RGB(0, 255, 255)	Cyan
vbWhite	RGB(255, 255, 255)	White

## Serialize

Serialized as a string.

### Syntax

```
BSTR Serialize();
```

### Return value

A string representing the FoxitWatermarkElementInfo.

### Remarks

Serializing the watermark as a string can be used to permanently save it.

## Foxit PhantomPDF.Creator

The Creator object that is used to convert non-PDF files to PDF.

### Methods

The Creator object has the following methods.

Method	Description
<a href="#">CombineFiles</a>	Combines multiple documents to a PDF.

## CombineFiles

Combines multiple documents to a PDF.

### Syntax

```
SHORT CombineFiles(BSTR bstrFiles, BSTR DestPDFFile, SHORT uCombineFlags);
```

### Parameters

bstrFiles	The source files.
-----------	-------------------

DestPDFFile	The destination PDF document path.
uCombineFlags	The flags, Enum CombineFlags

## Return value

The number of combined files, Less than zero if there are any errors.

## Remarks

The parameter bstrFiles can be either a folder path or a string of one or more file paths, with each separated by '|'.

The flag COMBINE\_TRAVEL\_SUBDIR of uCombineFlags can be ignored if bstrFiles is not a folder path.

# Foxit PhantomPDF.CombineFlags

An enum used for the "Creator.CombineFiles" parameter uCombineFlags.

## Constants

The Foxit PhantomPDF.CombineFlags enum has the following constants.

Constants	Value
COMBINE_DEFAULT	0x00
COMBINE_STOP_CONVERTFAIL	0x01
COMBINE_ADD_CONTENTS	0x02
COMBINE_TRAVEL_SUBDIR	0x04

## COMBINE\_DEFAULT

0x00, Default.

## Remarks

Skip the error file and continue. Don't build contents index page. Don't traverse subdirectory.

## COMBINE\_STOP\_CONVERTFAIL

0x01, Abort and exit if error occurred.

## COMBINE\_ADD\_CONTENTS

0x02, Add contents index to first page.

## COMBINE\_TRAVEL\_SUBDIR

0x04, Traverse subdirectory.

# Foxit PhantomPDF.PDSaveFlags

An enum used for the "PDDoc.Save's" parameter nType.

## Constants

The Foxit PhantomPDF.PDSaveFlags enum has the following constants.

Constants	Value
PDSaveIncremental	0x00
PDSaveFull	0x01
PDSaveCopy	0x02
PDSaveLinearized	0x04
PDSaveCollectGarbage	0x20

## PDSaveIncremental

Writes changes only.

## PDSaveFull

Writes the entire file.

## PDSaveCopy

Writes a copy of the file.

## PDSaveLinearized

Saves the file optimized for web viewing.

## PDSaveCollectGarbage

Remove unreferenced objects, often reducing file size.

# FoxitPDFCreatorForOffice

This section describes the OLE automation interface supported by the Foxit PhantomPDF Office add-in, which allows users to convert non-PDF files to PDF. The following table summarizes the available objects and data types.

Object	Description
<a href="#">FoxitPDFCreatorForOffice.PDFCreator</a>	The PDFCreator object is used to convert non-PDF files to PDF.
<a href="#">FoxitPDFCreatorForOffice.ISettings</a>	Conversion settings

## FoxitPDFCreatorForOffice.PDFCreator

The PDFCreator object is used to convert non-PDF files to PDF.

### Methods

The Creator object has the following methods.

Method	Description
<a href="#">ConvertToPDFEx</a>	Converts non-pdf to PDF.
<a href="#">GetCurrentConversionSettings</a>	Get parameter values from current conversion settings.

### ConvertToPDFEx

Converts non-PDF files already open within Microsoft Office to PDF.

### Syntax

```
HRESULT ConvertToPDF(IDispatch* pSettings, LONG* retVal);
```

### Parameters

pSettings	Converts settings.
retVal	Returns the result of execution.

### Return value

S\_OK if successful, S\_FAIL if not.

### GetCurrentConversionSettings

Get parameter values from current conversion settings.

## Syntax

```
HRESULT GetCurrentConversionSettings(IDispatch** retVal);
```

## Parameters

---

retVal	A Conversion Settings Object.
--------	-------------------------------

---

## Return value

S\_OK if successful, S\_FAIL if not.

# FoxitPDFCreatorForOffice. ISettings

Represents a data type of Conversion Settings.

## Properties

---

AddBookmarks	Whether to add bookmarks. The default value is true.
AddTags	Whether to add tags. The default value is true.
ConvertAllPages	Whether to convert all pages. The default value is true.
FitToOnePage	Whether to adjust the worksheet width so that the entire worksheet appears on one PDF page. The default value is false.
FitWidth	Whether to fit the worksheet to the width of a PDF page. The default value is false.
OutputPDFFileName	The output PDF document path. Can be null if user will decide the path.
PromptForPDFFilename	Whether to pop up the Save As dialog. Ignored if OutputPDFFileName is not null.
PromptForSheetSelection	Whether to pop up SheetSelection dialog. Ignored if PrintActivesheetOnly is False.
PrintActivesheetOnly	Only convert the active sheet. The default value is true.
ViewPDFFile	Run PhantomPDF and open the PDF file.

---

# Demo for VBA

## A simple VBA code skeleton of your application

```
Dim phApp As FoxitPhantomPDF.FoxitApp
Set phApp = CreateObject("FoxitExch.App")

Dim phAVDoc As FoxitPhantomPDF.FoxitAVDoc
Set phAVDoc = CreateObject("FoxitExch.AVDoc")

Call phAVDoc.Open("D:\TestDocument.pdf", test)

Dim bValidDoc As Boolean
bValidDoc = phAVDoc.IsValid

Dim PDDoc As FoxitPhantomPDF.FoxitPDDoc

If bValidDoc Then
Set PDDoc = phAVDoc.GetPDDoc
Call PDDoc.RotatePage(0, 1)
Call PDDoc.InsertPages(0, PDDoc, 1,1, 1)

'Do other things.....

PDDoc.Save PDSaveFull, "D:\TestFoxit.pdf"
PDDoc.Close
End If
phApp.Exit
```

## Adding a text watermark to the center of all pages



```
Dim PDDoc As FoxitPhantomPDF.FoxitPDDoc
Set PDDoc = CreateObject("FoxitExch.PDDoc")

PDDoc.Open ("D:\TestWatermark.pdf")

Dim phWmlInfo As FoxitPhantomPDF.FoxitWatermarkElementInfo
Set phWmlInfo = PDDoc.CreateWatermarkElementInfo()

phWmlInfo.Type = 0
phWmlInfo.WMText = "FoxitPhantomPDF"
phWmlInfo.FontName = "Helvetica"
phWmlInfo.FontSize = 24
phWmlInfo.TextColorRef = RGB(255, 0, 0)
phWmlInfo.Rotation = 0
phWmlInfo.Opacity = 0.5
phWmlInfo.WMScale = -1
phWmlInfo.Top = 1
phWmlInfo.VerticalDistance = 0
phWmlInfo.VerticalDistanceFrom = 1
phWmlInfo.HorizontalDistance = 0
phWmlInfo.HorizontalDistanceFrom = 1
phWmlInfo.Start = 0
phWmlInfo.End = -1
phWmlInfo.Even = True
phWmlInfo.Odd = True

PDDoc.AddWatermark phWmlInfo
PDDoc.Save PDSaveFull, "D:\AddTextWatermark.pdf"
```

## **Adding an image watermark to the center of all**

## pages

```
Dim PDDoc As FoxitPhantomPDF.FoxitPDDoc
Set PDDoc = CreateObject("FoxitExch.PDDoc")

PDDoc.Open ("D:\TestWatermark.pdf")

Dim phWmlInfo As FoxitPhantomPDF.FoxitWatermarkElementInfo
Set phWmlInfo = PDDoc.CreateWatermarkElementInfo()

phWmlInfo.Type = 1
phWmlInfo.WMFile = "D:\image.png"
phWmlInfo.WMFilePageIndex = 1
phWmlInfo.Rotation = 0
phWmlInfo.Opacity = 0.5
phWmlInfo.WMScale = -1
phWmlInfo.Top = 1
phWmlInfo.VerticalDistance = 0
phWmlInfo.VerticalDistanceFrom = 1
phWmlInfo.HorizontalDistance = 0
phWmlInfo.HorizontalDistanceFrom = 1
phWmlInfo.Start = 0
phWmlInfo.End = -1
phWmlInfo.Even = True
phWmlInfo.Odd = True

PDDoc.AddWatermark phWmlInfo
PDDoc.Save PDSaveFull, "D:\AddImageWatermark.pdf"
```

## Getting/Setting a PDF form's value

```
Dim PDDoc As FoxitPhantomPDF.FoxitPDDoc
Set PDDoc = CreateObject("FoxitExch.PDDoc")
```

```
Call PDDoc.Open("D:\TestFormDoc.pdf")
```

```
'Text Field
```

```
Dim strTextField0 As String
strTextField0 = PDDoc.GetFieldValue("Text Field0")
Debug.Print strTextField0
```

```
Call PDDoc.SetFieldValue("Text Field0", "Foxit PhantomPDF")
strTextField0 = PDDoc.GetFieldValue("Text Field0")
Debug.Print strTextField0
```

```
'Push Button
```

```
Dim strPushButton0 As String
strPushButton0 = PDDoc.GetFieldValue("Push Button0")
Debug.Print strPushButton0
```

```
Call PDDoc.SetFieldValue("Push Button0", "Test Button")
strPushButton0 = PDDoc.GetFieldValue("Push Button0")
Debug.Print strPushButton0
```

```
'Check Box
```

```
Dim strCheckBox0 As String
strCheckBox0 = PDDoc.GetFieldValue("Check Box0")
Debug.Print strCheckBox0
```

```
If strCheckBox0 = "Yes" Then
Call PDDoc.SetFieldValue("Check Box0", "Off")
```

```

Else
Call PDDoc.SetFieldValue("Check Box0", "Yes")
End If
strCheckBox0 = PDDoc.GetFieldValue("Check Box0")
Debug.Print strCheckBox0

'Radio Button
Dim strRadioButton0 As String
strRadioButton0 = PDDoc.GetFieldValue("Radio Button0")
Debug.Print strRadioButton0

If strRadioButton0 = "Yes" Then
Call PDDoc.SetFieldValue("Radio Button0", "Off")
Else
Call PDDoc.SetFieldValue("Radio Button0", "Yes")
End If
strRadioButton0 = PDDoc.GetFieldValue("Radio Button0")
Debug.Print strRadioButton0

'ComboBox
Dim strComboBox0 As String
strComboBox0 = PDDoc.GetFieldValue("Combo Box0")
Debug.Print strComboBox0

Call PDDoc.SetFieldValue("Combo Box0", " Combo Data1")
strComboBox0 = PDDoc.GetFieldValue("Combo Box0")
Debug.Print strComboBox0

Call PDDoc.SetFieldValue("Combo Box0", "out of term")
strComboBox0 = PDDoc.GetFieldValue("Combo Box0")

```

```
Debug.Print strComboBox0
```

```
'List Box
```

```
Dim strListBox0 As String
```

```
strListBox0 = PDDoc.GetFieldValue("List Box0")
```

```
Debug.Print strListBox0
```

```
Call PDDoc.SetFieldValue("List Box0", " List Data1 ")
```

```
strListBox0 = PDDoc.GetFieldValue("List Box0")
```

```
Debug.Print strListBox0
```

```
Call PDDoc.SetFieldValue("List Box0", "out of term")
```

```
strListBox0 = PDDoc.GetFieldValue("List Box0")
```

```
Debug.Print strListBox0
```

```
PDDoc.Save PDSaveFull, "D:\TestForm.pdf"
```

```
PDDoc.Close
```

## **OCR and export the document to an Excel document**

```
Dim PDDoc As FoxitPhantomPDF.FoxitPDDoc
```

```
Set PDDoc = CreateObject("FoxitExch.PDDoc")
```

```
Call PDDoc.Open("D:\ScanDocument.pdf")
```

```
Call PDDoc.OCRAndExportToExcel("D:\OCRAndExportToExcel_Text.xlsx", 1, 2, True, False)
```

```
Call PDDoc.OCRAndExportToExcel("D:\OCRAndExportToExcel_Image.xlsx", 1, 2, True, True)
```

## **Combining several documents to a PDF**

```
Dim phCreator As FoxitPhantomPDF.Creator
```

```
Set phCreator = CreateObject("FoxitExch.Creator")
```

```
Dim nCombinedCnt As Integer
```

```
nCombinedCnt = phCreator.CombineFiles("D:\image1.png|image2.png", "D:\combineFiles_files.pdf",  
COMBINE_ADD_CONTENTS)
```

```
Call phCreator.CombineFiles("D:\CombineFiles", "D:\combineFiles_folder.pdf",  
COMBINE_ADD_CONTENTS)
```

**Note** : The path "D:\CombineFiles" is a folder which contains several files that are supported.

## Converting non-PDF to PDF

```
Dim phCreator As FoxitPDFCreatorForOffice.pdfCreator
```

```
Dim phSetting As FoxitPDFCreatorForOffice.ISettings
```

```
Set phCreator = CreateObject("FoxitPDFCreatorForOffice.PDFCreator")
```

```
Dim pdfname As String
```

```
pdfname = "D:\TestConvert.pdf"
```

```
phCreator.GetCurrentConversionSettings phSetting
```

```
phSetting.AddBookmarks = True
```

```
phSetting.AddTags = True
```

```
phSetting.ConvertAllPages = False
```

```
phSetting.ViewPDFFile = True
```

```
phSetting.OutputPDFFileName = pdfname
```

```
phSetting.PromptForPDFFilename = False
```

```
phSetting.PromptForSheetSelection = True
```

```
phSetting.PrintActivesheetOnly = True
```

```
phCreator.CreatePDFEx phSetting, 0
```