# Kofax Power PDF

## Automation and Command Line Developer's Guide

Version: 5.1.0

Date: 2024-02-28

**TUNGSTEN**
**AUTOMATION**
**FORMERLY KOFAX**

# Table of Contents

# Preface

This guide provides instructions for developers on how to launch and manage Kofax Power PDF and its functionalities using command line parameters and automation interfaces.

## System requirements

The primary source of information about Kofax Power PDF 5.1.0 system requirements and dependencies on other products is the *Technical Specifications* document, which is available on the Product Documentation page. The document is updated regularly, and we recommend that you review it carefully to ensure success with your Kofax Power PDF product.

## Training

Kofax offers both classroom and online training to help you make the most of your product. To learn more about training courses and schedules, visit the Kofax Education Portal on the Kofax website.

## Getting help with Kofax products

The Kofax Knowledge Portal repository contains articles that are updated on a regular basis to keep you informed about Kofax products. We encourage you to use the Knowledge Portal to obtain answers to your product questions.

To access the Kofax Knowledge Portal, go to https://knowledge.kofax.com.

> 🛈 The Kofax Knowledge Portal is optimized for use with Google Chrome, Mozilla Firefox, or Microsoft Edge.

The Kofax Knowledge Portal provides:

- Powerful search capabilities to help you quickly locate the information you need.
  Type your search terms or phrase into the **Search** box, and then click the search icon.
- Product information, configuration details and documentation, including release news.

To locate articles, go to the Knowledge Portal home page and select the applicable Solution Family for your product, or click the View All Products button.

From the Knowledge Portal home page, you can:

- Access the Kofax Community (for all customers).
  On the Resources menu, click the **Community** link.

- Access the Kofax Customer Portal (for eligible customers).
  Go to the [Support Portal Information](#) page and click **Log in to the Customer Portal**.

- Access the Kofax Partner Portal (for eligible partners).
  Go to the [Support Portal Information](#) page and click **Log in to the Partner Portal**.

- Access Kofax support commitments, lifecycle policies, electronic fulfillment details, and self-service tools.
  Go to the [Support Details](#) page and select the appropriate article.

# Command line reference

This chapter describes command line parameters for Kofax Power PDF, for both 32-bit and 64-bit versions.

## Parameters for opening PDF files

Use the following parameters to call Kofax Power PDF (PowerPDF.exe) from the command line. See Command line examples for command line samples with short descriptions.

The default location of the executable may vary:

- The default path for Power PDF on a 32-bit version of Windows:
  `C:\Program Files\Kofax\Power PDF 51\bin\PowerPDF.exe`

- The default path for Power PDF on a 64-bit version of Windows:
  `C:\Program Files (x86)\Kofax\Power PDF 51\bin\PowerPDF.exe`

> ⓘ Parameters go into effect only if Power PDF is not yet running when it is called from the command line.

| Parameter | Extensional parameter | Effect |
|---|---|---|
| z0 | | Sets zoom to default. |
| z1 | | Sets zoom to Fit Page. |
| z2 | | Sets zoom to Fit Width. |
| z3 | | Sets zoom to Fit Height. |
| z4 | | Sets zoom to Fit Visible. |
| zs | Numeric; zoom level in percent | Sets zoom to the percent specified in the extensional parameter value. |
| pl1 | | Sets View/Scroll Options to Single Page. |
| pl2 | | Sets View/Scroll Options to Continuous. |
| pl4 | | Sets View/Scroll Options to Continuous Facing. |
| pl6 | | Sets View/Scroll Options to Facing. |
| pn | Numeric; page number | Displays the page specified in the extensional parameter value. |
| /p | Text; path with filename | Opens the PDF file specified in the parameter, prints it to the system default printer, then closes the PDF file. |
| /t | Text; path with filename<br>Text; printer name | Opens the PDF file specified in the first parameter, prints it to the printer defined in the second parameter, then closes the PDF file. |

# Command line examples

| Command with parameters | Description |
|---|---|
| `"C:\Program Files (x86)\Kofax\Power PDF 51\bin\PowerPDF.exe" /zs 150 "D:\test1.pdf"` | Calls Power PDF on a 64-bit version of Windows to open `D:\test1.pdf` with 150% zoom. |
| `"C:\Program Files (x86)\Kofax\Power PDF 51\bin\PowerPDF.exe" /pn 10 "D:\test1.pdf"` | Calls Power PDF on a 64-bit version of Windows to open `D:\test1.pdf` and go to page 10. |
| `" C:\Program Files\Kofax\Power PDF 51\bin\PowerPDF.exe" /z1 "D:\test1.pdf"` | Calls Power PDF on a 32-bit version of Windows to open `D:\test1.pdf` with Fit Page zoom. |

# Automation interfaces

This chapter describes automation interfaces for Kofax Power PDF, for both 32-bit and 64-bit versions.

## Overview

Power PDF provides the automation interfaces listed in the following table. These interfaces provide a wide range of functionality to control the application itself and to manage and manipulate PDF documents, along with their properties, pages, and annotations.

| Interface | Lifetime | Description |
|---|---|---|
| IApp | Creatable | Controls the Power PDF application. |
| IDDDoc | Creatable | Controls the PDF documents. |
| IDVDoc | Creatable | Controls the view of PDF documents. |
| IDDPage | Non-creatable | Accessible through the IDDDoc interface. Controls the pages within a PDF document. |
| IDDBookmark | Creatable | Designed to access bookmarks in PDF documents. |
| iDDAnnot | Non-creatable | Accessible through the IDDPage interface. |
| IDVPageView | Non-creatable | Accessible through the IDVDoc interface. Controls viewing PDF pages. |
| iHiliteList | Creatable | Designed to specify the location of highlighted contents within a PDF document. |
| iDDTextSelect | Creatable | Designed to set and show the highlights within a PDF document. |
| ActiveXIEHelper | Creatable | Facilitates the movement to various pages within a document and specifies various display and print choices. |
| iRect | Creatable | Designed to set or retrieve special properties of different objects. |
| ITime | Creatable | Designed to set or retrieve special properties of different objects. |
| IPoint | Creatable | Designed to set or retrieve special properties of different objects. |

This chapter provides full details of these interfaces and gives usage examples, along with implementation details and relationships between the different interfaces.

Methods are listed with their definition and input and return parameters, along with a short description of their usage and links to corresponding examples, which are listed in Code examples.

# IApp interface

The IApp interface is designed to drive the Power PDF application. It provides control over the application main window and its UI elements, ribbons, groups, tools, as well as drop-down menus and toolbars. The opened documents are also accessible through the interface.

The application main window can be hidden if the automation client performs document level operations only.

## Method Exit

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL Exit(); |
| **Parameters:** | None |
| **Return value:** | VARIANT_TRUE if powerpdf.exe exited properly. |
| **Remarks:** | Closes all opened documents and quits the application. |
| **Example:** | Example 1; Example 2 |

## Method CloseAllDocs

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL CloseAllDocs(); |
| **Parameters:** | None |
| **Return value:** | Closes all documents opened in the application. |
| **Remarks:** | Closes all documents referred by the IDVDoc interface pointer. The DDDoc objects must be closed by the Automation client application. |
| **Example:** | Example 2 |

## Method GetActiveDoc

| | |
|---|---|
| **Syntax:** | IDispatch* GetActiveDoc(); |
| **Parameters:** | None |
| **Return value:** | Retrieves an interface to the active DVDoc object. |
| **Remarks:** | Retrieves the topmost document selected by the user. |

## Method GetActiveTool

| | |
|---|---|
| **Syntax:** | BSTR GetActiveTool(); |
| **Parameters:** | None |
| **Return value:** | Returns the name of the selected tool. The return value can be a NULL pointer. |
| **Remarks:** | Retrieves the name of the selected tool. |

## Method GetDVDoc

| | |
|---|---|
| **Syntax**: | IDispatch* GetDVDoc(long nIndex); |
| **Parameters:** | long nIndex: The identifier for the DVDoc object. |
| **Return value:** | Retrieves an interface to the DVDoc object identified by its index. |
| **Remarks:** | The application object maintains a list of open PDF documents. You can get any of them by referring to their index. |

## Method GetLanguage

| | |
|---|---|
| **Syntax:** | BSTR GetLanguage(); |
| **Parameters:** | None |
| **Return value:** | Returns the abbreviated string of the language. The available language codes are the following: |

ARA | Arabic
CHT | Chinese Simplified
CHS | Chinese Traditional
CSY | Czech
DAN | Danish
NLD | Dutch
ENU | English
FIN | Finnish
FRA | French
DEU | German
HUN | Hungarian
ITA | Italian
JPN | Japanese
KOR | Korean
NOR | Norwegian
PLK | Polish
PTB | Portuguese (Brazil)
RUS | Russian
ESP | Spanish
SVE | Swedish
TRK | Turkish

| | |
|---|---|
| **Remarks:** | Identifies the product installation language. |

## Method GetNumDVDocs

| | |
|---|---|
| **Syntax:** | long GetNumDVDocs(); |
| **Parameters:** | None |
| **Return value:** | Returns the number of opened documents. |
| **Remarks:** | Use this method for the enumeration of open documents. |

## Method Hide

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL Hide(); |
| **Parameters:** | None |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Hides the application main window. |

## Method Show

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL Show(); |
| **Parameters:** | None |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Makes the application main window visible. |

## Method Minimize

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL Minimize(long bMinimize); |
| **Parameters:** | long bMinimize: If a positive number, the application is minimized. If 0, the normal state of the application is restored. |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Shows the application main window minimized. |

## Method Maximize

| | |
|---|---|
| **Syntax**: | VARIANT_BOOL Maximize(long bMaximize); |
| **Parameters:** | long bMaximize: If a positive number, the application is maximized. If 0, the normal state of the application is restored. |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Shows the application main window minimized. |

## Method Restore

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL Restore(long bRestore); |
| **Parameters:** | long bRestore: If a positive number, the application is restored to its original size and position and becomes active. |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Shows the application main window maximized. |

## Method SetActiveTool

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL SetActiveTool(BSTR szButtonName, long bPersistent); |
| **Parameters:** | BSTR szButtonName: The name of the tool.<br>long bPersistent: if a positive number, the tool remains active after it has been used; 0 otherwise. |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Activates a particular tool. |

## Method MenuItemExecute

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL MenuItemExecute(BSTR szMenuItemName); |
| **Parameters:** | BSTR szMenuItemName: Specifies the action to be performed. For a list of menu names, see the Appendix. |
| **Return value:** | VARIANT_TRUE if the action was started; VARIANT_FALSE otherwise. |
| **Remarks:** | Executes the action the same way as when the user presses the corresponding button or selects the application menu item. |

## Method MenuItemIsEnabled

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL MenuItemIsEnabled(BSTR szMenuItemName); |
| **Parameters:** | BSTR szMenuItemName: Specifies the action. For a list of menu names, see the Appendix. |
| **Return value:** | VARIANT_TRUE if action exists with this name, it is enabled and can be executed. Otherwise, the return value is VARIANT_FALSE. |
| **Remarks:** | Checks if a specific action is enabled and can be executed. |

## Method MenuItemIsMarked

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL MenuItemIsMarked(BSTR szMenuItemName); |
| **Parameters:** | BSTR szMenuItemName: Specifies the menu item. |
| **Return value:** | |
| **Remarks:** | Not supported in this release. |

## Method MenuItemRemove

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL MenuItemRemove(BSTR szMenuItemName); |
| **Parameters:** | BSTR szMenuItemName: Specifies the menu item. |
| **Return value:** | |
| **Remarks:** | Not supported in this release. |

## Method GetFrame

| | |
|---|---|
| **Syntax:** | IDispatch* GetFrame(); |
| **Parameters**: | None |
| **Return value:** | Returns a reference to a Rect object. |
| **Remarks:** | Retrieves the position and the size of the main application window. |

## Method SetFrame

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL SetFrame(IDispatch* iRect); |
| **Parameters:** | IDispatch* iRect: Dispatch interface of a Rect object that specifies the coordinates of the main application window. |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Specifies the coordinates of the main application window. |

## Method ToolButtonIsEnabled

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL ToolButtonIsEnabled(BSTR szButtonName); |
| **Parameters:** | BSTR szButtonName: Specifies the toolbar button. |
| **Return value:** | |
| **Remarks:** | Not supported in this release. |

## Method ToolButtonRemove

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL ToolButtonRemove(BSTR szButtonName); |
| **Parameters:** | BSTR szButtonName: Specifies the toolbar button. |
| **Return value:** | |
| **Remarks:** | Not supported in this release. |

## Method GetPreferenceEx

| | |
|---|---|
| **Syntax:** | VARIANT GetPreferenceEx(short nType); |
| **Parameters:** | short nType: Specifies the application preference item by its name. |
| **Return value:** | The value of the preference item. |
| **Remarks:** | Obtains the value of a specific preference item. |

## Method SetPreferenceEx

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL SetPreferenceEx (short nType, VARIANT * pVal); |
| **Parameters:** | short nType: Name of the preference item. |
| | VARIANT * pVal: Value to be set. |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Sets the value of a specific preference item. |

## Method Lock

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL Lock(BSTR szLockedBy); |
| **Parameters:** | BSTR szLockedBy: Used as the name of the application that locks Power PDF. |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Locks the application. After calling this method, UnlockEx must be called. Prevents users from using the user interface of Power PDF. Also prevents opening any documents in the application. |

## Method UnlockEx

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL UnlockEx(BSTR szLockedBy); |
| **Parameters**: | BSTR szLockedBy: Used as the name of the application that locked Power PDF. |
| **Return value**: | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Unlocks the application. Must be called if Lock was called previously. |

# IDDDoc interface

This interface controls PDF documents at the file level without involving user interface-related operations. The files can be opened, modified, saved, and closed. The interface provides a number of page operations within and between the documents.

## Method Create

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL Create(); |
| **Parameters:** | None |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Creates a new PDF document. The file does not contain any pages. |

## Method Open

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL Open(BSTR szFullPath); |
| **Parameters:** | BSTR szFullPath: Specifies the PDF file to be opened. |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Opens the specified PDF file. A document opened in this way is not displayed in the application window.<br>Note that the maximum number of open documents is 50. |
| **Example:** | Example 1 |

## Method Save

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL Save(short nType, BSTR szFullPath); |
| **Parameters:** | short nType: Specifies the method for how the document should be saved. The parameter can be a combination (with logical OR) of the following values: |
| | DDSaveIncremental (0): Write changes only, not the complete file. This always results in a larger file, even if objects have been deleted. |
| | DDSaveFull (1):  Write the entire file to the filename specified by szFullPath. |
| | DDSaveCopy (2): 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 DDSaveFull is also used. |
| | DDSaveCollectGarbage (32): Remove unreferenced objects; this often reduces the file size, and its usage is encouraged. This flag can only be specified if DDSaveFull is also used. |
| | DDSaveLinearized (4): Save the file optimized for the web, providing hint tables. This allows the PDF file to be byte-served. This flag can only be specified if DDSaveFull is also used. |
| | BSTR szFullPath: Specifies the file name. You are allowed to use the original file name that was used for the Open method. |
| **Return value:** | VARIANT_TRUE if the saving operation succeeded. |
| **Remarks:** | Saves the content of the document to the specified file. |
| **Example:** | Example 1 |

## Method Close

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL Close(); |
| **Parameters:** | None |
| **Return value:** | VARIANT_TRUE if the document was closed properly. |
| **Remarks:** | Closes the DDDoc object. |
| **Example:** | Example 1 |

## Method DeletePages

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL DeletePages(long nStartPage, long nEndPage); |
| **Parameters:** | long nStartPage: Specifies the first page to de deleted. |
| | long nEndPage: Specifies the last page to de deleted. |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Deletes the specified pages from the document. |

## Method GetFileName

| | |
|---|---|
| **Syntax:** | BSTR GetFileName(); |
| **Parameters:** | None |
| **Return value:** | Returns with the name of the file referenced by the DDDoc object. |
| **Remarks:** | Retrieves the name of the opened document. |

## Method MovePage

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL MovePage(long nMoveAfterThisPage, long nPageToMove); |
| **Parameters:** | long nMoveAfterThisPage: Specifies the page number after which the moved page is placed. |
| | long nPageToMove: Specifies the page to be moved. |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Moves a page to a new position within the document. |

## Method ReplacePages

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL ReplacePages(long nStartPage, IDispatch* iDDDocSource, long nStartSourcePage, long nNumPages, long bMergeTextAnnotations); |
| **Parameters:** | long nStartPage: Specifies the first page in the target document to be replaced. The next nNumPages pages are replaced. |
| | IDispatch* iDDDocSource: Specifies the source document from where the document pages are copied. |
| | long nStartSourcePage: Specifies the first page of the source document to be moved. |
| | long nNumPages: Specifies the number of pages to be moved to the target document. |
| | long bMergeTextAnnotations: Set it to value 1 to merge the annotations of the original and the new pages. In case of value 0, the text annotations are not merged. |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Replaces a given number of pages of the document with pages of another document. The annotations of the affected pages are optionally merged. |

## Method InsertPages

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL InsertPages(long nInsertPageAfter, IDispatch* iDDDocSource, long nStartPage, long nNumPages, long bBookmarks); |
| **Parameters:** | long nInsertPagesAfter: Specifies a page in the target document after which the new pages are not inserted. Use a value of -1 to insert pages in the beginning of the document. |
| | IDispatch* iDDDocSource: Specifies the source document from which the new pages are being inserted. |
| | long nStartPage: Specifies the first page of the source document that must be inserted into the target document. |
| | long nNumPages: Specifies the number of pages following the start page to be inserted. |
| | long bBookmarks: Set this parameter to 1 to copy the bookmarks along with the pages. Note that all bookmarks are copied, even if they do not point to existing pages in the new document. |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |

| | |
|---|---|
| **Remarks:** | Copies one or more pages from a PDF document and inserts them after the specified page in the current PDF file. |
| **Example:** | [Example 1](#) |

## Method GetFlags

| | |
|---|---|
| **Syntax:** | long GetFlags(); |
| **Parameters:** | None |
| **Return value:** | Returns the document flags. The return value is made by a logical OR operation of the following values. |
| | DDDocNeedsSave = 1:<br>The document has been modified. |
| | DDDocRequiresFullSave = 2:<br>Incremental saving would not be enough. Use the DDSaveFull flag when document is saved. |
| | DDDocIsModified = 4:<br>The document is modified (bookmarks or annotations have been added). |
| | DDDocDeleteOnClose = 8:<br>The document is temporary and must be deleted when the file is closed. |
| | DDDocWasRepaired = 16:<br>The original document was repaired during the method DDDoc.Open. The reason for the repair might be a damaged or incorrectly saved PDF file. |
| | DDDocIsLinearized = 1024:<br>The document is linearized. |
| | DDDocIsOptimized = 2048:<br>The document is optimized. |
| **Remarks:** | Retrieves the current state and other document parameters. |

## Method SetFlags

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL SetFlags(long nFlags); |
| **Parameters:** | long nFlags: The required value of the document flags. |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Sets the document flags. The possible values are described under the method GetFlags. Most of the flags are read-only and cannot be set or cleared. DDDocNeedsSave and DDDocDeleteOnClose flags can be set and cleared. |

## Method ClearFlags

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL ClearFlags(long nFlags); |
| **Parameters:** | long nFlags: Specify the document flags to be cleared. |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Clears document flags. Most flags are read-only and cannot be set or cleared. DDDocNeedsSave and DDDocDeleteOnClose flags can be set and cleared. |

## Method GetPageMode

| | |
|---|---|
| **Syntax:** | long GetPageMode(); |
| **Parameters:** | None |
| **Return value:** | The return value can be one the following:<br>DDDontCare = 0,<br>DDUseNone = 1,<br>DDUseThumbs = 2,<br>DDUseBookmarks = 3,<br>DDFullScreen = 4 |
| **Remarks:** | Retrieves the current page mode of the document. This property controls how the document is displayed initially: displays only the document pages, the pages and the thumbnails, or the pages and the bookmarks. The document can also be displayed in full screen mode. |

## Method SetPageMode

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL SetPageMode(long nPageMode); |
| **Parameters:** | long nPageMode: Specifies the required Page Mode. |
| **Return value:** | VARIANT_TRUE if the Page Mode was set properly. |
| **Remarks:** | Sets the Page Mode of the document. The possible values of the Page Mode property are described under the method GetPageMode. |

## Method OpenDVDoc

| | |
|---|---|
| **Syntax:** | IDispatch* OpenDVDoc(BSTR szTitle); |
| **Parameters:** | BSTR szTitle: The title of the document view. |
| **Return value:** | Returns a DVDoc object referring to the DDDoc object. |
| **Remarks:** | Creates a DVDoc object associated to the DDDoc object. A new document view is displayed that shows the document referred by the DDDoc object. This call makes the application visible if it was hidden originally. |

## Method DeleteThumbs

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL DeleteThumbs(long nStartPage, long nEndPage); |
| **Parameters:** | long nStartPage, nEndPage: These two values specify the interval of pages. The thumbnails of these pages are removed from the file. |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Removes embedded thumbnail images of the pages. |

## Method CreateThumbs

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL CreateThumbs(long nFirstPage, long nLastPage); |
| **Parameters:** | long nFirstPage, nLastPage: These two values specify the interval of pages. The thumbnails for these pages are added to the file. |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Creates embedded thumbnail images of the pages or a range of pages. |

## Method GetInfo

| | |
|---|---|
| **Syntax:** | BSTR GetInfo(BSTR szInfoKey); |
| **Parameters:** | BSTR szInfoKey: Specifies the key whose value is required. |
| **Return value:** | Retrieves the value of the specified key. |
| **Remarks:** | Retrieves the document information values such as Author and Title. |

## Method SetInfo

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL SetInfo(BSTR szInfoKey, BSTR szBuffer); |
| **Parameters:** | BSTR szInfoKey: Specifies the key whose value must be set. <br><br> BSTR szBuffer: Specifies the value. |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Sets the different document information parameters, such as Author and Title. |

## Method GetPermanentID

| | |
|---|---|
| **Syntax:** | BSTR GetPermanentID(); |
| **Parameters:** | None |
| **Return value:** | Returns the permanent ID of the document. |

## Method GetInstanceID

| | |
|---|---|
| **Syntax:** | BSTR GetInstanceID(); |
| **Parameters:** | None |
| **Return value:** | Returns the instance ID of the document. |

## Method CropPages

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL CropPages(long nStartPage, long nEndPage, short nEvenOrOddPagesOnly, IDispatch* iRect); |
| **Parameters:** | long nStartPage: Specifies the first page to be cropped.<br><br>long nEndPage: Specifies the last page to be cropped.<br><br>short nEvenOrOddPagesOnly:<br>0 – Crop all pages in the range<br>1 – Crop odd pages in the range<br>2 – Crop even pages in the range<br><br>IDispatch* iRect: Specifies the cropping rectangle. The coordinates should be given in user space. |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Crops the pages in the given page range. |

## Method AcquirePage

| | |
|---|---|
| **Syntax:** | IDispatch* AcquirePage(long nPage); |
| **Parameters:** | long nPage: Specifies the index of the page to be acquired. |
| **Return value:** | Returns a reference to the specified PDF page. |
| **Remarks:** | Retrieves a reference to a document page. The reference can be used for different page operations. The reference is the IDispatch interface of a DDPage object. |
| **Example:** | Example 2 ; Example 3; |

## Method CreateTextSelect

| | |
|---|---|
| **Syntax:** | IDispatch* CreateTextSelect(long nPage, IDispatch* iRect); |
| **Parameters:** | long nPage: Specifies the page where the selection should be created.<br><br>IDispatch* iRect: Specifies the area of the text selection. |
| **Return value:** | Returns a reference to a text selection object. |
| **Remarks:** | Creates a text selection based on a specific area of a page. |

## Method GetNumPages

| | |
|---|---|
| **Syntax:** | long GetNumPages(); |
| **Parameters:** | None |
| **Return value:** | Returns the number of pages in the document. |

## Method GetJSObject

| | |
|---|---|
| **Syntax:** | IDispatch GetJSObject(); |
| **Parameters:** | None |
| **Return value:** | Returns the JavaScript object for this document. |
| **Remarks:** | With the use of the JavaScript Object, runs JavaScript code. |
| **Example:** | Example 5; |

# IDVDoc interface

This interface is closely related to the DDDoc interface. While DDDoc provides the file-level operations, this interface supplies methods to support viewing documents. Similar to the other interface, you can open and close documents using DVDoc methods, but all of them control the document window and make that visible along with the application main window. The strong relationship between IDVDoc and DDDoc is represented with the method of GetDDoc, which gives access to file level operations on the document opened in the document window.

## Method GetDDDoc

| | |
|---|---|
| **Syntax:** | IDispatch* GetDDDoc() |
| **Parameters:** | None |
| **Return value:** | Returns the IDispatch interface of the associated DDDoc object. |
| **Remarks:** | Allows access to the DDDoc object of the DVDoc object. You may need this method to reach the properties of the DDDoc object or call its methods. |
| **Example:** | Example 2 ; Example 3; |

## Method Open

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL Open(BSTR szFullPath); |
| **Parameters:** | BSTR szFullPath: Specifies the PDF file or a document type that can be converted to PDF in Power PDF. |
| **Return value:** | VARIANT_TRUE if the file was opened. |
| **Remarks:** | Opens the specified file and displays it in a child window of the main application window. |
| **Example:** | Example 2 ; Example 3; |

## Method Close

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL Close(long bNoSave); |
| **Parameters:** | long bNoSave: Set it to 1 if you do not want to save the file; 0 value means that the application prompts for saving modifications. |
| **Return value:** | VARIANT_TRUE if the file was closed properly. |
| **Remarks:** | This method closes the document along with its window. |

## Method PrintPages

| Syntax: | VARIANT_BOOL PrintPages(long nFirstPage, long nLastPage, long nPSLevel, long bBinaryOk, long bShrinkToFit); |
|---|---|
| Parameters: | nFirstPage: The first page to be printed. It is zero-based.<br>nLastPage: The last page to be printed.<br>nPSLevel: If 2, PostScript Level 2 operators are used; if 3, PostScript Language Level 3 operators are also used.<br>bBinaryOk: If 0, all data is encoded as 7-bit ASCII; if a positive number, binary data may be included in the PostScript program.<br>bShrinkToFit: If a positive number, the page is shrunk to fit within the usable area of the printed page; if 0, it is not. |
| Return value: | VARIANT_TRUE if the operation succeeded. |
| Remarks: | Displays the Print dialog and prints the specified range of pages. |

## Method PrintPagesSilent

| Syntax: | VARIANT_BOOL PrintPagesSilent(long nFirstPage, long nLastPage, long nPSLevel, long bBinaryOk, long bShrinkToFit); |
|---|---|
| Parameters: | nFirstPage: The first page to be printed. It is zero-based.<br>nLastPage: The last page to be printed.<br>nPSLevel: If 2, PostScript Level 2 operators are used; if 3, PostScript Language Level 3 operators are also used.<br>bBinaryOk: If 0, all data is encoded as 7-bit ASCII; if a positive number, binary data may be included in the PostScript program.<br>bShrinkToFit: If a positive number, the page is shrunk to fit within the usable area of the printed page; if 0, it is not. |
| Return value: | VARIANT_TRUE if the operation succeeded. |
| Remarks: | Prints the specified ranges of pages without displaying the Print dialog box. |

## Method GetTitle

| Syntax: | BSTR GetTitle(); |
|---|---|
| Parameters: | None |
| Return value: | Returns the title of the window associated with the document. |

## Method Maximize

| Syntax: | VARIANT_BOOL Maximize(long bMaxSize); |
|---|---|
| Parameters: | long bMaxSize: If the value of the input parameter is greater than zero, the document window is within the application main window. |
| Return value: | VARIANT_TRUE if the operation succeeded. |
| Remarks: | Maximizes the size of the document window associated to the DVDoc object. |

## Method GetFrame

| | |
|---|---|
| **Syntax:** | IDispatch* GetFrame(); |
| **Parameters:** | None |
| **Return value:** | Returns the IDispatch interface of a Rect object specifying the coordinates of the window associated with the document. |
| **Remarks:** | The returned IDispatch interface refers to the CPowerPDFRect object that contains the coordinates of the document window. |

## Method IsValid

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL IsValid(); |
| **Parameters:** | None |
| **Return value:** | VARIANT_TRUE if the document is a valid PDF document. |

## Method GetViewMode

| | |
|---|---|
| **Syntax:** | long GetViewMode(); |
| **Parameters:** | None |
| **Return value:** | The return value indicates how the document is displayed. |
| **Remarks:** | The return values can be the following:<br>DDDontCare:0 – Leave the view mode as it is<br>DDUseNone:1 – Display without bookmarks or thumbnails<br>DDUseThumbs:2 – Display using thumbnails<br>DDUseBookmarks:3 – Display using bookmarks<br>DDFullScreen:4 – Display in full screen mode |

## Method SetViewMode

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL SetViewMode(long nType); |
| **Parameters:** | long nType |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Modifies the view of the document. The possible values are listed under the GetViewMode method. |

## Method GetDVPageView

| | |
|---|---|
| **Syntax:** | IDispatch* GetDVPageView(); |
| **Parameters:** | None |
| **Return value:** | The method returns the IDispatch interface of the DVPageView object. |
| **Remarks:** | Allows access to methods in the IDVPageView interface. |

## Method SetTextSelection

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL SetTextSelection(IDispatch* iDDTextSelect); |
| **Parameters:** | IDispatch* iDDTextSelect: Reference to an iDDTextSelect object that specifies the text selection in the document. |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Sets the required text selection. |
| **Example:** | Example 3; |

## Method ShowTextSelect

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL ShowTextSelect(); |
| **Parameters:** | None |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Shows the selected text. |
| **Example:** | Example 3; |

## Method ClearSelection

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL ClearSelection(); |
| **Parameters:** | None |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Clears the text selection. |

## Method BringToFront

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL BringToFront(); |
| **Parameters:** | None |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Makes the document window topmost among other document windows within the application window. This window gets the focus. |

## Method SetTitle

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL SetTitle(BSTR szTitle); |
| **Parameters:** | BSTR szTitle: Specifies the new title of the document window. |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Changes the title of the document window. |

## Method PrintPagesEx

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL PrintPagesEx(long nFirstPage, long nLastPage, long nPSLevel, long bBinaryOk, long bShrinkToFit, long bReverse, long bFarEastFontOpt, long bEmitHalftones, long iPageOption); |
| **Parameters:** | nFirstPage: The first page to be printed. It is zero-based. |
| | nLastPage: The last page to be printed. |
| | nPSLevel: If 2, PostScript Level 2 operators are used; if 3, PostScript Language Level 3 operators are also used. |
| | bBinaryOk: If 0, all data is encoded as 7-bit ASCII; if a positive number, binary data can be included in the PostScript program. |
| | bShrinkToFit: If a positive number, the page is shrunk to fit within the usable area of the printed page; if 0, it is not. |
| | bReverse: (PostScript printing only) If a positive number, print the pages in reverse order. If 0, print the pages in the regular order. |
| | bFarEastFontOpt: (PostScript printing only) Set to a positive number if the destination printer has multi-byte fonts; set to 0 otherwise. |
| | bEmitHalftones: (PostScript printing only) If a positive number, emit the halftones specified in the document. If 0, do not. |
| | iPageOption: Pages in the range to print. Must be one of: DDAllPages (-3), DDEvenPagesOnly (-4), or DDOddPagesOnly (-5). |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Displays the Print dialog box and prints the specified pages. |

## Method PrintPagesSilentEx

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL PrintPagesSilentEx(long nFirstPage, long nLastPage, long nPSLevel, long bBinaryOk, long bShrinkToFit, long bReverse, long bFarEastFontOpt, long bEmitHalftones, long iPageOption); |
| **Parameters:** | long nFirstPage: The first page to be printed. It is zero-based. |
| | long nLastPage: The last page to be printed. |
| | long nPSLevel: If 2, PostScript Level 2 operators are used; if 3, PostScript Language Level 3 operators are also used. |
| | long bBinaryOk: If 0, all data is encoded as 7-bit ASCII; if a positive number, binary data may be included in the PostScript program. |
| | long bShrinkToFit: If a positive number, the page is shrunk to fit within the usable area of the printed page; if 0, it is not. |

<table>
<tr><td></td><td>long bReverse: (PostScript printing only) If a positive number, print the pages in reverse order. If 0, print the pages in the regular order.</td></tr>
</table>

|  |  |
|---|---|
|  | long bReverse: (PostScript printing only) If a positive number, print the pages in reverse order. If 0, print the pages in the regular order. |
|  | long bFarEastFontOpt: (PostScript printing only) Set to a positive number if the destination printer has multi-byte fonts; set to 0 otherwise. |
|  | long bEmitHalftones: (PostScript printing only) If a positive number, emit the halftones specified in the document. If 0, do not. |
|  | long iPageOption: Pages in the range to print. Must be one of: DDAllPages (-3), DDEvenPagesOnly (-4), or DDOddPagesOnly(-5). |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Prints the specified pages silently without displaying the Print dialog box. |

## Method GetParentDDDoc

| **Syntax:** | IDispatch* GetParentDDDoc(); |
|---|---|
| **Parameters:** | None |
| **Return value:** | Returns the IDispatch interface of a Doc object or NULL if the function fails. |
| **Remarks:** | Returns a reference to the parent document if the document is inside a package; NULL otherwise. |

## Method FindText

| **Syntax:** | VARIANT_BOOL FindText(BSTR szText, long bCaseSensitive, long bWholeWordOnly, long bReset); |
|---|---|
| **Parameters:** | BSTR szText: Text to be found. |
|  | long bCaseSensitive: 0 – Not case-sensitive; positive number – case-sensitive. |
|  | long bWholeWordOnly: 0 – Only whole words are matched; positive number – partial words are matched. |
|  | long bReset: 0 – Search starts from current page; positive number – search starts from the beginning of the document. |
| **Return value:** | VARIANT_TRUE if text is found. |
| **Remarks:** | Finds text in the document and make it visible by scrolling the document to the text. The text is highlighted. |

## Method OpenInWindowEx

| **Syntax:** | VARIANT_BOOL OpenInWindowEx(BSTR szFullPath, long hWnd, long openFlags, long useOpenParams, long pgNum, short pageMode, short |
|---|---|

|  |  |
|---|---|
|  | zoomType, long zoom, short top, short left); |
| **Parameters:** | BSTR szFullPath: Specifies the PDF file to be opened. |
|  | long hWnd: Handle for the window in which the file is displayed |
|  | long openFlags:<br>AV_EXTERNAL_VIEW – Display the pageview, scrollbars, toolbars, and bookmark pane.<br>AV_DOC_VIEW – Display the pageview, scrollbars, and bookmark pane<br>AV_PAGE_VIEW – Display only the pageview. |
|  | long useOpenParams: 0 – Document defined open action is used; positive number – open action is defined in methods' parameters. |
|  | long pgNum: page number to be opened. Valid if useOpenParams is positive. |
|  | short pageMode:<br>DDDontCare = 0 – Leave the mode as it is<br>DDUseNone = 1 – Display without bookmark<br>DDUseThumbs = 2 – Display using thumbnails<br>DDUseBookmarks = 3 – Display usingbookmarks<br>DDFullScreen = 4 – Display in full screen mode |
|  | short zoomType:<br>DVZoomNoVary (0)        A fixed zoom, such as 75%<br>DVZoomFitPage (1)        Fits the page in the window<br>DVZoomFitWidth (2)        Fits the page width to the window<br>DVZoomFitHeight (3)        Fits the page height to the window<br>DVZoomFitVisibleWidth (4)   Fits the visible content of the page width into the window |
|  | long zoom: Zoom factor, used only for DVZoomNoVary if useOpenParams is a positive number. |
|  | short top: Currently not used |
|  | short left: Currently not used |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Opens a file in a user-specified window. |

# IDDPage Interface

This interface represents a page of a PDF document. Certain page level operations – cropping and rotating pages – and managing annotations and text highlights can be performed by its methods. The interface can be retrieved by the method AcquirePage of the DDDoc interface.

## Method GetRotate

| | |
|---|---|
| **Syntax:** | short GetRotate(); |
| **Parameters:** | None |
| **Return value:** | The method returns the rotation value of the page in degrees. The possible values are listed under the method SetRotate. |
| **Remarks:** | Determines the rotation degree of the page. |

## Method SetRotate

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL SetRotate(short nRotate); |
| **Parameters:** | short nRotate: Rotates the page with the given degree. The acceptable values are 0, 90, 180, and 270. |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Influences the view of the document page. To make the changes permanent, the PDF file must be saved. |

## Method GetNumAnnots

| | |
|---|---|
| **Syntax:** | long GetNumAnnots(); |
| **Parameters:** | None |
| **Return value:** | Returns the number of annotations on the given page. |
| **Remarks:** | Typically used when the annotations must be enumerated. |
| **Example:** | Example 4; |

## Method RemoveAnnot

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL RemoveAnnot(long nIndex); |
| **Parameters:** | long nIndex: Specifies the index of the annotation to be deleted from the page. First annotation has 0 index. |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Removes specified annotation from the page. |

## Method GetNumber

| | |
|---|---|
| **Syntax:** | long GetNumber(); |
| **Parameters:** | None |
| **Return value:** | Retrieves the index of the page. Note that the page indexes start from zero. |
| **Remarks:** | Certain methods expect a page index as the input parameter. Use this method to retrieve the appropriate value for the given page. |

## Method CropPage

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL CropPage(IDispatch* iRect); |
| **Parameters:** | IDispatch* iRect: The IDispatch interface to a Rect obcject that determines the size of the crop area of the page. |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Adjusts the visible area of the page. The coordinates must be given in the user space. |

## Method GetAnnot

| | |
|---|---|
| **Syntax:** | IDispatch* GetAnnot(long nIndex); |
| **Parameters:** | long nIndex: Specifies the index of the requested annotation on the page. |
| **Return value:** | The method retrieves the IDispatch interface of the selected annotation (DDAnnot) object. |
| **Remarks:** | Retrieves the given page annotation. The properties of the annotation can be accessed through the returned interface pointer. |

## Method AddAnnot

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL AddAnnot(long nIndexAddAfter, IDispatch* iDDAnnot); |
| **Parameters:** | long nIndexAddAfter: Specifies the index of the new annotation. The annotation is inserted after the specified index. Use a value of -1 in case of insertion at the very first position or if the given annotation is the first one on the page. |
| | IDispatch* iDDAnnot: The reference to the annotation to be inserted. Note that the DDAnnot object is not a creatable one. You can get this interface pointer using other methods of the IDDPage interface. |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Adds an annotation to the page. Typical usage of the method is to copy or move existing annotations to another page. The annotation can also be retrieved from another document. To create a new annotation, use the method AddNewAnnot. |

## Method AddNewAnnot

| | |
|---|---|
| **Syntax:** | IDispatch* AddNewAnnot(long nIndexAddAfter, BSTR szSubType, IDispatch* iRect); |
| **Parameters:** | long nIndexAddAfter: Specifies the index of the new annotation. The annotation is inserted after the specified index. Use a value of -1 in case of insertion at the very first position or if the given annotation is the first one on the page.<br><br>BSTR szSubType: Specifies the type of annotation. It supports the "Text" subtype only.<br><br>IDispatch* iRect: The IDispatch interface of a Rect object specifies the location and the size of the annotation. |
| **Return value:** | The method returns the IDispatch interface of the new DDAnnot object if the insertion succeeded. You can use this value to set additional properties of the annotation (such as color). The return value is NULL if the operation failed. |
| **Remarks:** | Creates a new annotation on the given page. |
| **Example:** | Example 3; |

## Method GetDoc

| | |
|---|---|
| **Syntax:** | IDispatch* GetDoc(); |
| **Parameters:** | None |
| **Return value:** | Returns the IDispatch interface of the DDDoc object associated to the DDPage object. |

## Method GetAnnotIndex

| | |
|---|---|
| **Syntax:** | long GetAnnotIndex(IDispatch* iDDAnnot); |
| **Parameters:** | IDispatch* iDDAnnot: The IDispatch interface of the DDAnnot object whose index must be retrieved. |
| **Return value:** | Returns the zero-based index of the given annotation. |

## Method GetSize

| | |
|---|---|
| **Syntax:** | IDispatch* GetSize(); |
| **Parameters:** | None |
| **Return value:** | Returns the IDispatch interface of a Point object. |
| **Remarks:** | Retrieves the size of the page measured in points. |

## Method CreateWordHilite

| | |
|---|---|
| **Syntax:** | IDispatch* CreateWordHilite(IDispatch* iHiliteList); |
| **Parameters:** | IDispatch* iHiliteList: Specifies the series of characters to be highlighted. |
| **Return value:** | The method returns the IDispatch interface of a DDTextSelect object. |
| **Remarks:** | Highlights a series of words on the given page. The DDHiliteList object may specify several series of words. |
| **Example:** | Example 2, Example 2 (code) |

## Method CreatePageHilite

| | |
|---|---|
| **Syntax:** | IDispatch* CreatePageHilite(IDispatch* iHiliteList); |
| **Parameters:** | IDispatch* iHiliteList: Specifies the series of characters to be highlighted. |
| **Return value:** | The method returns the IDispatch interface of a DDTextSelect object. |
| **Remarks:** | Highlights a series of characters on the given page. The DDHiliteList object may specify several series of characters. |
| **Example:** | The example provided for the method CreateWordHilite is applicable with the appropriate changes. See Example 3. |

## Method DrawEx

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL DrawEx ( long window, long displayContext, IDispatch * updateRect, short nxOrigin, short nyOrigin, short zoom ); |
| **Parameters:** | long window: Handle for the window into which the content is to be drawn |
| | long displayContext: Not used, leave it zero. |
| | IDispatch * updateRect: Reference to the area of the page in user space coordinates to be drawn. Any content outside the area is not drawn. |
| | short nxOrigin: X coordinate of the part of the page to be drawn. |
| | short nyOrigin: Y coordinate of the part of the page to be drawn. |
| | short zoom: Zoom value with which page content will be drawn as a percentage. 100 means magnification 1. |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Draws the specified part of the page into a user-specified window. |

## Method CopyToClipboard

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL CopyToClipboard ( IDispatch * boundRect, short nXOrigin, short nYOrigin, short nZoom ); |
| **Parameters:** | IDispatch * boundRect: Reference to the bounding rect in device space coordinates to be drawn. Any content outside the area will not be drawn.<br><br>short xOrigin:X coordinate of the part of the page to be drawn.<br><br>short yOrigin: Y coordinate of the part of the page to be drawn.<br><br>short zoom: Zoom value with which page content will be drawn as a percentage. 100 means magnification 1. |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Copies page content as an image to the clipboard. This method is only available on 32-bit systems. |

# IDDBookmark interface

This interface provides access to bookmarks in documents. New bookmarks cannot be created with the OLE Automation interface but can be permanently removed. Basically, bookmarks are designed to navigate to specified areas of a document.

## Method Destroy

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL Destroy ( ); |
| **Parameters:** | None |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Removes bookmark from the document. |

## Method GetByTitle

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL GetByTitle (IDispatch *IDDDoc, BSTR bookmarkTitle ); |
| **Parameters:** | IDispatch * IDDDoc: Reference to the document object to get the bookmark from.<br><br>BSTR bookmarkTitle: Title of the bookmark to get. Title is case-sensitive. |
| **Return value:** | VARIANT_TRUE if the bookmark exists with the given title and the operation succeeded. |
| **Remarks:** | Gets bookmark by its title. The bookmark is not the return value of the document. The object's reference is set to the bookmark instead. |

## Method GetTitle

| | |
|---|---|
| **Syntax:** | BSTR GetTitle ( ); |
| **Parameters:** | None |
| **Return value:** | Returns the title of the bookmark object. |

## Method IsValid

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL IsValid ( ); |
| **Parameters:** | None |
| **Return value:** | VARIANT_TRUE if the bookmark is valid. |
| **Remarks:** | It does not make a thorough test on the bookmark's data structure. |

## Method Perform

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL Perform ( IDispatch * IDVDoc ); |
| **Parameters:** | IDispatch * IDVDoc: Reference to the document object where the bookmark exists. |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Performs an action of the bookmark. |

## Method SetTitle

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL SetTitle (_bstr_t szNewTitle ); |
| **Parameters:** | None |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Sets the title of the bookmark. |

# iDDAnnot interface

This interface provides access to properties of the annotations, such as color, date, title, content, coordinates measured in user space, open state, and subtype. Because the list of annotations is maintained by the DDPage object, the interface is accessible by method DDPage::GetAnnot. The annotations are identified by their index within the given page. You can also add annotations to a page using the DDPage object. Please note that only annotations with a "Text" subtype are creatable.

## Method GetColor

| | |
|---|---|
| **Syntax:** | long GetColor(); |
| **Parameters:** | None |
| **Return value:** | Returns the color of the annotation in RGB values. The values are represented in this form: 0xBBGGRR. So the value of 0xFF (255) represents a light red color, 0xFF00 (65280) is a light green value, and 0xFF0000 (16711680) is light blue. |

## Method SetColor

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL SetColor(long nRGBColor); |
| **Parameters:** | long nRGBColor: Specifies the color of the annotation |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Example:** | Example 4 |

## Method GetDate

| | |
|---|---|
| **Syntax:** | IDispatch* GetDate(); |
| **Parameters:** | None |
| **Return value:** | The method returns an IDispatch interface to a Time object. |
| **Remarks:** | Uses the properties of the Time object to retrieve the date of the annotation. |

## Method SetDate

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL SetDate(IDispatch* ITime); |
| **Parameters:** | IDispatch* ITime: Specifies the creation time of the annotation. |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |

## Method GetRect

| | |
|---|---|
| **Syntax:** | IDispatch* GetRect(); |
| **Parameters:** | None |
| **Return value:** | Returns the IDispatch interface for the Rect object. |
| **Remarks:** | Returns the bounding rectangle of the annotation. Use the properties of the Rect object to determine the location of the annotation on the page. |

## Method SetRect

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL SetRect(IDispatch* iRect); |
| **Parameters:** | None |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Sets or changes the location and size of the annotation. |

## Method GetTitle

| | |
|---|---|
| **Syntax:** | BSTR GetTitle(); |
| **Parameters:** | None |
| **Return value:** | Returns the title of the annotation. |

## Method SetTitle

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL SetTitle(BSTR szTitle); |
| **Parameters:** | |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Sets or changes the title of the annotation. |

## Method GetSubType

| | |
|---|---|
| **Syntax:** | BSTR GetSubtype(); |
| **Parameters:** | None |
| **Return value:** | Returns the subtype property of the annotation. |

## Method IsValid

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL IsValid(); |
| **Parameters:** | None |
| **Return value:** | VARIANT_TRUE if the annotation is valid. |
| **Remarks:** | Verifies the annotation object and checks whether it is deleted. |

## Method IsOpen

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL IsOpen(); |
| **Parameters:** | None |
| **Return value:** | VARIANT_TRUE if the text annotation is open. |
| **Remarks:** | Verifies whether a text annotation is open. |

## Method SetOpen

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL SetOpen(long bIsOpen); |
| **Parameters:** | None |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Opens a text annotation. |

## Method GetContents

| | |
|---|---|
| **Syntax:** | BSTR GetContents(); |
| **Parameters:** | None |
| **Return value:** | Retrieves the content of an annotation. |

## Method SetContents

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL SetContents(BSTR szContents); |
| **Parameters:** | None |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Sets or modifies the content of the annotation. |
| **Example:** | Example 3; |

## Method IsEqual

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL IsEqual(IDispatch* iDDAnnot); |
| **Parameters:** | None |
| **Return value:** | VARIANT_TRUE if the two compared annotations are equal. |
| **Remarks:** | Determines whether the current annotation is the same as a specified one. |

## Method Perform

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL Perform(IDispatch* IDVDoc); |
| **Parameters:** | None |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Refers to Link type annotations and performs the action associated with the Link object. |

# IDVPageView

Similar to the DVDoc interface, this interface provides a set of methods related to viewing PDF pages. The interface is not creatable, but accessible via the method GetDVPageView in the DVDoc interface.

## Method DoGoBack

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL DoGoBack(); |
| **Parameters:** | None |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | The application tracks the views displayed in each document window. Use this method to step back to the previous view of the document window. |

## Method DoGoForward

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL DoGoForward(); |
| **Parameters:** | None |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | The application tracks the views displayed in each document window. Use this method to step forward to the next view of the document window. |

## Method GetDVDoc

| | |
|---|---|
| **Syntax:** | IDispatch* GetDVDoc(); |
| **Parameters:** | None |
| **Return value:** | Returns the IDispatch interface of the DVDoc object displayed in the page view. |
| **Remarks:** | Retrieves the DVDoc object displayed in the view. |

## Method GetDoc

| | |
|---|---|
| **Syntax:** | IDispatch* GetDoc(); |
| **Parameters:** | None |
| **Return value:** | Returns the IDispatch interface of the DDDoc object displayed in the page view. |
| **Remarks:** | Retrieves the DDDoc object displayed in the view. |

## Method GetPage

| | |
|---|---|
| **Syntax:** | IDispatch* GetPage(); |
| **Parameters:** | None |
| **Return value:** | Returns the IDispatch interface of the DDPage object referring to the current PDF page displayed in the page view. |
| **Remarks:** | Retrieves the DDPage object displayed in the view. |

## Method GetPageNum

| | |
|---|---|
| **Syntax:** | long GetPageNum(); |
| **Parameters:** | None |
| **Return value:** | Returns the zero-based index of the current PDF page displayed in the page view. |
| **Remarks:** | Retrieves the index of the displayed page. |

## Method GetZoom

| | |
|---|---|
| **Syntax:** | long GetZoom(); |
| **Parameters:** | None |
| **Return value:** | Returns the current zoom factor as a percentage. |

## Method GetZoomType

| | |
|---|---|
| **Syntax:** | short GetZoomType(); |
| **Parameters:** | None |
| **Return value:** | The method returns a given zoom type. |
| **Remarks:** | The possible return values are the following:<br>DVZoomNoVary (0)　　　　　A fixed zoom, such as 75%.<br>DVZoomFitPage (1)　　　　Fits the page in the window.<br>DVZoomFitWidth (2)　　　　Fits the page width to the window.<br>DVZoomFitHeight (3)　　　Fits the page height to the window.<br>DVZoomFitVisibleWidth (4)　Fits the visible content of the page width into the window. |

## Method Goto

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL Goto(long nPage); |
| **Parameters:** | long nPage: Zero-based index of the page |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Makes a page specified with its index visible. |

## Method ReadPageDown

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL ReadPageDown( ); |
| **Parameters:** | None |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Scrolls the document down by one screen area. |

## Method ReadPageUp

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL ReadPageUp(long nPage); |
| **Parameters:** | None |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Scrolls the document up by one screen area. |

## Method ScrollTo

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL ScrollTo(short nX, short nY); |
| **Parameters:** | short nX, short nY: Specifies the scrolling coordinates within the current page. |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Scrolls the current page to the given position. The coordinates specify the left top corner of the visible area in user space. |

## Method ZoomTo

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL ZoomTo(short nType, short nScale); |
| **Parameters:** | short nType: Specifies the Zoom type (see also the method GetZoomType)<br>short nScale: Specifies the scale of zoom as a percentage. |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |

## Method DevicePointToPage

| | |
|---|---|
| **Syntax:** | IDispatch* DevicePointToPage(IDispatch* IPoint); |
| **Parameters:** | IDispatch* IPoint: Point to be converted. |
| **Return value:** | The method returns the converted values. |
| **Remarks:** | Conversion from device space to user space. |

## Method PointToDevice

| Syntax: | IDispatch* PointToDevice(IDispatch* IPoint); |
|---|---|
| **Parameters:** | IDispatch* IPoint: Point to be converted. |
| **Return value:** | The method returns the converted values. |
| **Remarks:** | Conversion from user space to device space. |

## Method GetAperture

| Syntax: | IDispatch* GetAperture(); |
|---|---|
| **Parameters:** | None |
| **Return value:** | The method returns the IDispatch interface of a Rect object. |
| **Remarks:** | The Rect object specifies the visible area of the page in user space. |

# iHiliteList

The HiliteList object is designed to specify parts of the text within a PDF document to be selected. The elements of the list may refer to individual characters or words depending on the usage of the object.

## Method Add

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL Add(short nOffset, short nLength); |
| **Parameters:** | short nOffset: Specifies the index of the first object (character or word) to be highlighted.<br>short nLength: Specifies the number of objects to be highlighted. |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | The specified selections refer to characters if the HiliteList object is used by method CreatePageHilite of the DDPage object. They refer to words in the case of the method CreateWordHilite. |

# iDDTextSelect

The interface provides some read-only properties for the text selection on a given PDF page.

## Method Destroy

| | |
|---|---|
| **Syntax:** | VARIANT_BOOL Destroy(); |
| **Parameters:** | None |
| **Return value:** | VARIANT_TRUE if the operation succeeded. |
| **Remarks:** | Deletes the current text selection. |

## Method GetBoundingRect

| | |
|---|---|
| **Syntax:** | IDispatch* GetBoundingRect(); |
| **Parameters:** | None |
| **Return value:** | The method returns a Rect object. |
| **Remarks:** | Provides the bounding rectangle of the current text selection. |

## Method GetNumText

| | |
|---|---|
| **Syntax:** | long GetNumText(); |
| **Parameters:** | None |
| **Return value:** | Returns the number of selected text objects. |

## Method GetPage

| | |
|---|---|
| **Syntax:** | long GetPage(); |
| **Parameters:** | None |
| **Return value:** | Returns the index of the page where the selection is being placed. |

## Method GetText

| | |
|---|---|
| **Syntax:** | BSTR GetText(long nTextIndex); |
| **Parameters:** | long nTextIndex: Specifies the index of the selected text. |
| **Return value:** | Returns the selected text. |
| **Remarks:** | Retrieves the text specified with the given index in the highlighted content. |

# Helper interfaces

The following interfaces are designed for retrieving special properties of different objects.

## iRect interface

This interface is designed to get coordinates of a rectangular object. The interface provides four properties for the left, right, top, and bottom coordinates. The type of these properties is long. You have full access (read/write) to all four properties. See Example 4.

## ITime interface

This interface defines a Time object with seven properties: year, month, day, hour, minute, second, and millisecond. You have full access (read/write) to all four properties.

## IPoint interface

This interface defines a point with two properties: the X and Y coordinates. You have full access (read/write) to these properties. This interface is also used for referring to the size of an object. In this case, the property X refers to the width and Y to the height of the object.

# ActiveXIEHelper

The ActiveXIEHelper object comprises a set of methods that provide access to PDF browser options. This a creatable interface, and it facilitates the movement to various pages within a document, and specifies various display and print choices.

## Method GoBackwardStack

| | |
|---|---|
| **Syntax:** | GoBackwardStack(); |
| **Parameters:** | None |
| **Remarks:** | Checks the previous view on the view stack if a previous view exists. This view can be part of a different document. |

## Method GoForwardStack

| | |
|---|---|
| **Syntax:** | GoForwardStack(); |
| **Parameters:** | None |
| **Remarks:** | Checks the next view on the view stack if the next view exists. This view can be part of a different document. |

## Method GotoFirstPage

| | |
|---|---|
| **Syntax:** | gotoFirstPage(); |
| **Parameters:** | None |
| **Remarks:** | Shifts to the first page of the document, keeping the present location within the page and the present zoom level. |

## Method GotoLastPage

| | |
|---|---|
| **Syntax:** | gotoLastPage(); |
| **Parameters:** | None |
| **Remarks:** | Shifts to the last page of the document, keeping the existing location within the page and the existing zoom level. |

## Method GotoNextPage

| | |
|---|---|
| **Syntax:** | gotoNextPage(); |
| **Parameters:** | None |
| **Remarks:** | Moves to the next page in the document if the next page exists. Sustains the existing location within the page and the existing zoom level. |

## Method GotoPreviousPage

| | |
|---|---|
| **Syntax:** | gotoPreviousPage(); |
| **Parameters:** | None |
| **Remarks:** | Moves to the previous page in the document if the previous page exists. Sustains the existing location within the page and the existing zoom level. |

## Method Print

| | |
|---|---|
| **Syntax:** | Print(); |
| **Parameters:** | None |
| **Remarks:** | Prints the document as per the selected option in the user dialog box. Printing options comprise embedded printing (printing within a bounding rectangle on a given page) and interactive printing to a defined printer. The method returns instantly, even if printing is still in process.<br><br>**Note**: If the security settings restrict printing, the document is not printed. In this case the method is ignored. |

## Method PrintAll

| | |
|---|---|
| **Syntax:** | printAll(); |
| **Parameters:** | None |
| **Remarks:** | Prints the complete document without displaying the user dialog box. The present printer, page settings, and job settings are used while printing. The method returns instantly, even if printing is still in process.<br><br>**Note**: If the security settings restrict printing, the document is not printed. In this case the method is ignored. |

## Method PrintAllFit

| | |
|---|---|
| **Syntax:** | printAllFit(VARIANT_BOOL bOn); |
| **Parameters:** | bOn: Determines whether to scale the printable area while printing the document. Zero value of the parameter indicates that scaling is not required, and if the parameter has a positive value, it indicates that pages are scaled, if required, to fit into the printable area of the page of the printer. |
| **Remarks:** | Prints the complete document without displaying the user dialog box, and the pages are scaled, if required, to fit into the printable area of the page of the printer. The present printer, page settings, and job settings are used while printing. The method returns instantly, even if printing is still in process.<br><br>**Note**: If the security settings restrict printing, the document is not |

printed. In this case the method is ignored.

## Method PrintPages

| | |
|---|---|
| **Syntax:** | printPages( Long nFrom, Long nTo); |
| **Parameters:** | nFrom: Specifies the page number of the first page to be printed. Zero is the first page of the document. |
| | nTo: Specifies the page number of the last page to be printed. |
| **Remarks:** | Prints the defined pages without displaying the user dialog box. The present printer, page settings, and job settings are used while printing. The method returns instantly, even if printing is still in process. |
| | **Note**: If the security settings restrict printing, the document is not printed. In this case the method is ignored. |

## Method PrintPagesFit

| | |
|---|---|
| **Syntax:** | printPagesFit( Long nFrom, Long nTo, VARIANT_BOOL bShrinkToFit); |
| **Parameters:** | nFrom: Specifies the page number of the first page to be printed. Zero is the first page of the document. |
| | nTo: Specifies the page number of the last page to be printed. |
| | bShrinkToFit: Defines whether the pages will be scaled, if required, to fit into the printable area of a page of the printer. |
| **Remarks:** | Prints the required pages without displaying the user dialog box. The present printer, page settings, and job settings are used while printing. A parameter defines whether to scale pages, if required. The method returns instantly, even if printing is still in process. |
| | **Note**: If the security settings restrict printing, the document is not printed. In this case the method is ignored. |

## Method PrintWithDialog

| | |
|---|---|
| **Syntax:** | printWithDialog(); |
| **Parameters:** | None |
| **Remarks:** | Prints the document as per the selected options in the user dialog box. Printing options comprise of embedded printing (that printing within a bounding rectangle on a given page) and interactive printing to a defined printer. The method returns instantly, even if printing is still in process. |
| | **Note**: If the security settings restrict printing, the document is not printed. In this case the method is ignored. |

## Method SetCurrentPage

| | |
|---|---|
| **Syntax:** | setCurrentPage(Long nPage); |
| **Parameters:** | nPage: Specifies the page number of the target page. Zero page is the first page of the document. |
| **Remarks:** | Moves to the defined page in the document. Keep the existing location within the page and the existing zoom level. |

## Method SetLayoutMode

| | |
|---|---|
| **Syntax:** | setLayoutMode(BSTR szLayoutMode); |
| **Parameters:** | szLayoutMode - Possible values: |
| | Don't Care: Displays the present user preference. |
| | Single Page: Displays single page mode. |
| | OneColumn: Displays one-column continuous mode. |
| | TwoColumnLeft: Displays the two-column continuous mode with the first page on the left. |
| | TwoColumnRight: Displays the two-column mode with the first page on the right. |
| **Remarks:** | Displays the layout mode for the page view as per the string defined. |

## Method SetNamedDest

| | |
|---|---|
| **Syntax:** | setNamedDest(BSTR szNamedDest); |
| **Parameters:** | szNamedDest: The defined destination where the user goes. |
| **Remarks:** | Modifies the page view to the defined destination in the specified string. |

## Method SetPageMode

| | |
|---|---|
| **Syntax:** | setPageMode(BSTR szPageMode); |
| **Parameters:** | szPageMode - Possible values: |
| | None: Shows the document but by default; the bookmarks and the thumbnails are not displayed. |
| | Bookmarks: Shows the documents along with the bookmarks. |
| | Thumbs: Shows the document along with the thumbnails. |
| **Remarks:** | Defines the page mode as per the specified string. |

## Method SetShowToolbar

| | |
|---|---|
| **Syntax:** | setShowToolbar(VARIANT_BOOL bOn); |
| **Parameters:** | bOn: The positive value of the parameter signifies that the toolbar is displayed, and zero value indicates that it is not displayed. |
| **Remarks:** | Defines whether the toolbar will be displayed in the viewer. |

## Method SetView

| | |
|---|---|
| **Syntax:** | setView(BSTR szVviewMode); |
| **Parameters:** | szViewMode - Possible values:<br><br>Fit: Fits the complete page within the frame both vertically and horizontally.<br><br>FitH: Fits the complete width of the page within the frame.<br><br>FitV: Fits the complete height of the page within the frame.<br><br>FitB: Fits the bounding box within the frame both vertically and horizontally.<br><br>FitBH: Fits the complete width of the bounding box within the frame.<br><br>FitB: Fits the complete height of the bounding box within the frame. |
| **Return value:** | None |
| **Remarks:** | Defines the view of the page as per the specified string. |

## Method SetViewRect

| | |
|---|---|
| **Syntax:** | setViewRect(FLOAT left, FLOAT top, FLOAT width, FLOAT height); |
| **Parameters:** | left: The horizontal coordinate in the upper left.<br><br>Top: The vertical coordinate in the upper left corner.<br><br>Width: The horizontal width of the rectangle.<br><br>Height: The vertical height of the rectangle. |
| **Remarks:** | Specifies the rectangle view as per the defined coordinates. |

## Method SetViewScroll

| | |
|---|---|
| **Syntax:** | setViewRect(BSTR szViewMode, FLOAT offset); |
| **Parameters:** | szViewMode – Possible values:<br><br>Fit: Fits the complete page within the frame both vertically and horizontally.<br><br>FitH: Fits the complete width of the page within the frame<br><br>FitV: Fits the complete height of the page within the frame. |

| | |
|---|---|
| | FitB: Fits the bounding box within the frame both vertically and horizontally. |
| | FitBH: Fits the complete width of the bounding box within the frame. |
| | FitBV: Fits the complete height of the bounding box within the frame. |
| | Offset: The vertical or the horizontal coordinate placed either at the left or the top edge. |
| **Remarks:** | Defines the page view as per the defined string. Depending on the view of the page, the page is scrolled to the right or down as per the value specified for offset. |

## Method SetZoom

| | |
|---|---|
| **Syntax:** | setZoom(FLOAT percent); |
| **Parameters:** | percent: The required zoom factor, defined in percentage. For example, 0.7 represents a magnification of 70%. |
| **Remarks:** | Defines the magnification as per the specified value. |

## Method SetZoomScroll

| | |
|---|---|
| **Syntax:** | setZoomScroll(FLOAT percent, FLOAT left, FLOAT top); |
| **Parameters:** | percent: The required zoom factor, defined in percentage. For example, 0.7 represents a magnification of 70%. |
| | left: the horizontal coordinate placed at the left edge. |
| | Top: the vertical coordinate placed at the top edge. |
| **Remarks:** | Defines the magnification as per the specified value, and scrolls the page view both vertically and horizontally as per the defined values. |

## Property Src

| | |
|---|---|
| **Syntax:** | [get/set] src |
| **Return value:** | The property returns the URL of the document formatted as a string. |
| **Remarks:** | Receives or sets the URL for the document. |

# Chapter 3

# Code examples

These examples are based on Visual C++ and Visual Basic code. The examples are easily applicable to other programming languages that support OLE automation.

## Example 1 – Inserting pages into a document

The following example demonstrates how pages can be inserted from one PDF document into another. The sample can also be applied to the methods ReplacePages and Move Pages of the IDDDoc interface.

### Referenced methods

Creating Application interface

Creating DDDoc interface

IDDDoc::Open

IDDDoc:: InsertPages

IDDDoc::Save

IDDDoc::Close

IApp::Exit

### C++

```cpp
HRESULT Example1(LPWSTR sourceFile, LPWSTR targetFile, LPWSTR resultFile)

{

        PowerPDF::IDDDocPtr ddDocTarget;

        ddDocTarget.CreateInstance(__uuidof(PowerPDF::DDDoc));


        if (ddDocTarget->Open(targetFile) != VARIANT_TRUE)

        {

                pApp->Exit();
```

```cpp
        return E_FAIL;

}


/*------------------------

Other file types e.g. Office formats can be opened as PDF

if( 0 == ddDocTarget->Open(L"myworddoc.docx") )

{

pApp->Exit();

return E_FAIL;

}

-------------------------*/


PowerPDF::IDDDocPtr ddDocSource;

ddDocSource.CreateInstance(__uuidof(PowerPDF::DDDoc));


if (ddDocSource->Open(sourceFile) != VARIANT_TRUE)

{

        ddDocTarget->Close();

        pApp->Exit();

        return E_FAIL;

}


// Insert the first two pages of the source document into the 3rd position

// of the target document

if (ddDocTarget->InsertPages(1, ddDocSource, 0, 2, FALSE) != VARIANT_TRUE)

{

        ddDocTarget->Close();

        ddDocSource->Close();

        pApp->Exit();

        return E_FAIL;

}
```

```cpp
        // Save the modifications using DDDoc object
        if (ddDocTarget->Save(PowerPDF::DDSaveFull, resultFile) != VARIANT_TRUE)
        {
            //Saving the file failed
            return ERROR_SAVE;
        }
        ddDocTarget->Close();
        ddDocSource->Close();
        pApp->Exit();
        return S_OK;
}
```

## VB

```vb
Sub  Example1(inFile, outFile, resFile)


Set PDFApp = CreateObject("PowerPDF.App")
Set ddDocTarget = CreateObject("PowerPDF.DDDoc")


If ddDocTarget.Open(outFile) = False Then
    PDFApp.Exit
    Exit Sub
End If


Set ddDocSource = CreateObject("PowerPDF.DDDoc")
If ddDocSource.Open(inFile) = False Then
    ddDocTarget.Close
    PDFApp.Exit
    Exit Sub
End If


If ddDocTarget.InsertPages(1, ddDocSource, 0, 2, False) = False Then
    ddDocSource.Close
    ddDocTarget.Close
```

```
    PDFApp.Exit

    Exit Sub

End If


If ddDocTarget.Save(1, resFile) = False Then    '1 = DDSaveFull

    'Saving the file failed

    InputBox("Saving the file failed")

End If


ddDocSource.Close

ddDocTarget.Close

PDFApp.Exit

End Sub
```

# Example 2 – Highlighting words in a PDF document

This sample demonstrates how words can be selected within a page of a PDF document.

## Referenced methods

Creating Application interface

Creating DVDoc interface

Creating HiliteList interface

IDVDoc::Open

IDDDoc::GetDDDoc

DDDoc::AcquirePage

iHiliteList::Add

DDPage::CreateWordHilite

DVDoc::SetTextSelection

DVDoc::ShowTextSelection


## C++

```
HRESULT Example2(LPWSTR file)
```

```cpp
{
    //Open the PDF document
    PowerPDF::IDVDocPtr dvOpenDoc;
    dvOpenDoc.CreateInstance(__uuidof(PowerPDF::DVDoc));
    if (dvOpenDoc->Open(file) != VARIANT_TRUE)
    {
        //"ERROR: can't open test file\n"
        pApp->Exit();
        return E_FAIL;
    }


    //Get the associated DDDoc object
    PowerPDF::IDDDocPtr ddDoc = dvOpenDoc->GetDDDoc();


    //Acquire the page to be highlighted
    PowerPDF::IDDPagePtr ddPage = ddDoc->AcquirePage(0);


    //Create the list of highlights
    PowerPDF::iHiliteListPtr hl;
    hl.CreateInstance(__uuidof(PowerPDF::HiliteList));
    hl->Add(0, 1);      //      1st word should be highlighted
    hl->Add(2, 2);      //      and 2 additional words from the 3rd position
    hl->Add(5, 3);      //      and 3 additional words from the 6th position
    hl->Add(9, 4);      //      and 4 additional words from the 10th position


                        //Create the word highlights and display them
    PowerPDF::iDDTextSelectPtr ts = ddPage->CreateWordHilite(hl);
    dvOpenDoc->SetTextSelection(ts);
    dvOpenDoc->ShowTextSelect();
    //      The highlight list may refer to series of characters as well.
    //      Use method ddPage->CreatePageHilite to select series of
    //      characters
```

```
        return S_OK;
}
```

## VB

```vb
dim Offset
dim Length
Offset = array(0,2,5,9)
Length = array(1,2,3,4)
hNum = 4


'''''''''

Call Example2("C:\Test\MyDoc.pdf", 0, Offset, Length, hNum)
'''''''''


Sub  Example2(file, pg, offs, len, num)


Set PDFApp = CreateObject("PowerPDF.App")
Set dvOpenDoc = CreateObject("PowerPDF.DVDoc")


If dvOpenDoc.Open(file) = False Then
   PDFApp.Exit
   Exit Sub
End If


Set ddDoc = dvOpenDoc.GetDDDoc()
Set ddPage = ddDoc.AcquirePage(pg)
Set hl = CreateObject("PowerPDF.HiliteList")


for i = 0 to num -1
   hr = hl.Add(offs(i), len(i))
next
```

```
Set ts = ddPage.CreateWordHilite(hl)

hr = dvOpenDoc.SetTextSelection(ts)

dvOpenDoc.ShowTextSelect

End Sub
```

# Example 3 – Adding annotations

This sample provides a code for adding a new annotation to a given page of the document and setting its properties.

## Referenced methods

Creating Application interface

Creating DVDoc interface

DDPage::AcquirePage

Rect properties

DDPage::GetNumAnnots

DDPage::AddNewAnnot

DDAnnot::SetContents

DDAnnot::SetColor

## C++

```cpp
HRESULT AppendTextAnnotation(PowerPDF::IDDDocPtr ddDocOpen, long pageindex, RECT rc,
LPCTSTR strComment)

{

    // Acquiring the page of the document

    PowerPDF::IDDPagePtr ddPage = ddDocOpen->AcquirePage(pageindex);

    if (NULL == ddPage)

    {

        return ERROR_INVALID_PAGEINDEX;

    }


    PowerPDF::iRectPtr rcAnnot;

    if (S_OK != rcAnnot.CreateInstance(__uuidof(PowerPDF::Rect)))

    {

        ddPage.Release();

        return ERROR_INTERNAL;

    }


    rcAnnot->Putleft(rc.left);
```

```cpp
        rcAnnot->Puttop(rc.top);

        rcAnnot->PutRight(rc.right);

        rcAnnot->PutBottom(rc.bottom);

        long index = ddPage->GetNumAnnots() - 1;


        PowerPDF::iDDAnnotPtr ddAnnot = ddPage->AddNewAnnot(index, _T("Text"), rcAnnot);

        if (NULL == ddAnnot)

        {

            ddPage.Release();

            return ERROR_ADD_ANNOT_FAILED;

        }


        if (ddAnnot->SetContents(strComment) != VARIANT_TRUE)

        {

            ddPage.Release();

            ddAnnot.Release();

            return ERROR_SETCONTENT;

        }


        ddAnnot->SetColor(128);

        return S_OK;

}
```

### VB

```vb
dim rect
' (Left,Top,Right,Bottom)
rect=array(176,0,418,550)


'''''''''''''''''''''''''

Call AppendTextAnnotation("C:\Test\MyDoc.pdf",0,rect,"This is an annotation")
'''''''''''''''''''''''''


Sub  AppendTextAnnotation(file,pg,rc,str)
```

```
Set PDFApp = CreateObject("PowerPDF.App")

Set dvOpenDoc = CreateObject("PowerPDF.DVDoc")


If dvOpenDoc.Open(file) = False Then

    PDFApp.Exit

    Exit Sub

End If


Set ddDoc = dvOpenDoc.GetDDDoc()

Set ddPage = ddDoc.AcquirePage(pg)

Set rcAnnot = CreateObject("PowerPDF.Rect")


rcAnnot.Bottom = rc(3)

rcAnnot.Left = rc(0)

rcAnnot.Right = rc(2)

rcAnnot.Top = rc(1)


index = ddPage.GetNumAnnots - 1

Set ddAnnot = ddPage.AddNewAnnot(index, "Text", rcAnnot)

ddAnnot.SetContents (str)

ddAnnot.SetColor (128)


End Sub
```

# Example 4 – Opening PDF in browser with OLE automation

This example contains both the style (CSS) and JavaScript blocks to define a PDFObject container and embed the content of a sample document.

## Prerequisites

Internet Explorer 11

## C++

```cpp
HRESULT Example4(LPWSTR htmlFile)

{

        // Open the PDF document

        PowerPDF::IDVDocPtr dvOpenDoc;

        dvOpenDoc.CreateInstance(__uuidof(PowerPDF::DVDoc));

        if (dvOpenDoc->Open(htmlFile) != VARIANT_TRUE)

        {

                pApp->Exit();

                return E_FAIL;

        }

        return(S_OK);

}
```

## VB

```vb
Sub Example4(htmlFile)


Set PDFApp = CreateObject("PowerPDF.App")

Set dvOpenDoc = CreateObject("PowerPDF.DVDoc")

If dvOpenDoc.Open(htmlFile) = False Then

    PDFApp.Exit

End If

End Sub
```

# Example 5 – Form field access via JavaScript object

This sample shows how to get access to the JavaScript object to run JavaScript code.

It opens two documents and shows how to access the opened documents and how to get/set form field values in the documents.

## VB

```vb
Set PDFApp = CreateObject("PowerPDF.App")

Set dvOpenDoc = CreateObject("PowerPDF.DVDoc")

dvOpenDoc.Open("C:\Test\Empty.pdf")

If dvOpenDoc.Open("C:\Test\Form_sample_with_Text1_textfield.pdf") = False Then

    PDFApp.exit()

End If

Set ddDoc = dvOpenDoc.GetDDDoc()

Set jso = ddDoc.GetJSObject

docs = jso.app.activeDocs


jso.app.alert ("First opened document: '" & docs(0).title & "', Last opened document: '" & docs(1).title &"'")


sField = docs(0).getField("Text1")

If Not IsNull(sField) Then

        jso.app.alert ("Found 'Text1' field in '"& docs(0).title & "' document")

Else

        jso.app.alert ("There is no 'Text1' field in '"& docs(0).title & "' document")

End If


sField = docs(1).getField("Text1")

If Not IsNull(sField) Then

        jso.app.alert ("The old value for 'Text1' field in '"& docs(1).title & "' is: " &
docs(1).getField("Text1").value)

        docs(1).getField("Text1").value = "New Value for 'Text1' form field"

        jso.app.alert ("The new value for 'Text1' field in '"& docs(1).title & "' is: " &
docs(1).getField("Text1").value)

Else
```

```
        jso.app.alert ("There is no 'Text1' field in '"& docs(1).title & "' document")
End If


sField = docs(1).getField("Updated")
If Not IsNull(sField) Then
        docs(1).getField("Updated").value = "yes"
Else
        jso.app.alert ("There is no 'Updated' field in '"& docs(1).title & "' document")
End If


ddDoc.Save 1, "C:\Test\Form_sample_with_Text1_textfield_newValue.pdf"  '1 = DDSaveFull
dvOpenDoc.Close (1)
PDFApp.exit()
```

# Example 6 – Unattended file comparison

Calling the Compare command on the Advanced Processing tab requires user interaction when saving the resulting document. For an unattended file comparison, use the hidden AutoCompare control instead.

Open the documents and specify the path and file name for the resulting output file, then call jso.app.execMenuItem("AutoCompare") to perform the comparison.

Call the dvOpenDoc.Open method to load files. For example:

> dvOpenDoc.Open("c:\temp\DocToCompare1.pdf")

> dvOpenDoc.Open("c:\temp\DocToCompare2.pdf")

Call the ddNewDoc.Save method to save the resulting document. For example:

> ddNewDoc.Save 1,"c:\temp\ComparisonResult.pdf"

### VB

```
Set PDFApp = CreateObject("PowerPDF.App")


Set dvOpenDoc = CreateObject("PowerPDF.DVDoc")
dvOpenDoc.Open("c:\temp\DocToCompare1.pdf")
Set dvOpenDoc2 = CreateObject("PowerPDF.DVDoc")
dvOpenDoc2.Open("c:\temp\DocToCompare2.pdf ")


Set ddDoc = dvOpenDoc.GetDDDoc()
Set jso = ddDoc.GetJSObject
jso.app.execMenuItem("AutoCompare")


Set dvNewDoc = PDFApp.GetActiveDoc()
Set ddNewDoc = dvNewDoc.GetDDDoc()
ddNewDoc.Save 1,"c:\temp\ComparisonResult.pdf"
PDFApp.exit()
```

# Appendix

## Action and menu names

You can use these names as input parameters for the methods <u>MenuItemExecute</u> and <u>MenuItemIsEnabled</u> of the IApp interface.

### File menu

| Action | Menu name |
|--------|-----------|
| Info / Description / Properties | DocumentProperties |
| Open / Browse | Open |
| New / Blank PDF | CreateDocumentFromBlankPage |
| New / Form file / Create a PDF from a single file | CreateFromFile |
| New / Form file / Create PDFs from multiple files | CreateMultipleFiles |
| New / Form file / Combine multiple files into a single PDF | CombineMultipleFiles |
| New / Form file / Overlay multiple files into a single PDF | OverlayMultipleFiles |
| New / From clipboard | CreateDocumentFromClipboard |
| New / Create PDF Portfolio | CreatePortfolio |
| New / From Scanner / Create new PDF from scanner | ScanNew |
| New / From Scanner / Append new scan to current file | ScanExist |
| New / From Scanner / Scanner Setup | ScanSetting |
| New / From Web Page | NewFromWeb |
| Close | Close |
| Save | Save |
| Save As | SaveAs |
| Share / E-mail | Email |
| Share / Fax via e-mail | FaxViaEmail |
| Print | Print |
| Options | Preferences |
| Exit | Exit |

## Home tab

| Action | Menu name |
| --- | --- |
| Pages / Insert | InsertPage |
| Pages / Extract | ExtractPage |
| Pages / Delete | DeletePage |
| *No applicable button | ReplacePage |
| *No applicable button | AddPage |
| Pages / Document Assembly | DocumentAssembly |
| Create / From File | CreateFromFile |
| Create / Combine Files | CombineMultipleFiles |
| Create / Combine All | CombinAllOpenFile |
| Create / From Scanner / Scan New | ScanNew |
| Create / From Scanner / Scan To Existing | ScanExist |
| Create / From Scanner / Scan And Markup | ScanMarkup |
| Create / From Scanner / Scan Settings | ScanSetting |
| Page Rotate / Right | RotatePageRight |
| Page Rotate / Left | RotatePageLeft |
| Page Rotate / Advanced | RotatePage |
| Convert / MS Word | SaveAsWord |
| Convert / Excel | SaveAsExcel |
| Convert / PowerPoint | SaveAsPPT |
| Convert / Other / XPS Document | SaveAsXPS |
| Convert / Other / Word Form | SaveAsForm |
| Convert / Other / WordPerfect Document | SaveAsWPD |
| Convert / Other / MRC PDF | SaveAsMRC |
| Convert / Other / Searchable PDF | SaveAsSearchable |
| Convert / Make PDF Searchable / Make PDF Searchable | MakeSearchable |
| Convert / Make PDF Searchable / Proofreader | ProofReader |
| Tools / Edit Text | EditText |
| Tools / Typewriter | Typewriter |
| Tools / Reduce / Optimize | Optimize |
| Tools / Reduce / Reduce current file | ReduceCurrentFile |
| Tools / Reduce / Reduce multiple files | ReduceMultipleFiles |
| Tools / Split | Split |

| Action | Menu name |
|--------|-----------|
| Tools / Sign & Mark | FillSign |
| Search / Search / Search Current | Find |
| Search / Search / Search Multiple | Search |

## Edit tab

| Action | Menu name |
|--------|-----------|
| Convert / Convert to editable | ConvertToEditable |
| Basic / Undo | Undo |
| Basic / Redo | Redo |
| Basic / Delete | Delete |
| Basic / Select Area | SelectArea |
| Basic / Select All | SelectAll |
| Basic / Deselect All | DeselectAll |
| Clipboard / Cut | Cut |
| Clipboard / Copy | Copy |
| Clipboard / Paste | Paste |
| Modify / Crop | Crop |
| Modify / Edit Text | EditText |
| Modify / Edit Object | EditObject |
| Modify / Typewriter | Typewriter |
| Insert / Headers and Footers / Add… | AddHeadersAndFooters |
| Insert / Headers and Footers / Update… | UpdateHeadersAndFooters |
| Insert / Headers and Footers / Remove… | RemoveHeadersAndFooters |
| Insert / Bates Numbering / Add… | AddBatesNumbers |
| Insert / Bates Numbering / Remove… | RemoveBatesNumbers |
| Insert / Link | Link |
| Insert / Movie | Movie |
| Insert / Sound | Sound |
| Insert / 3D | 3D |
| Tools / Alignment / View Grid | ViewGrid |
| Tools / Alignment / Snap to Grid | SnapToGrid |
| Tools / Alignment / View Rulers | ViewRulers |
| Tools / Alignment / View Guides | ViewGuides |

| Action | Menu name |
|---|---|
| Tools / Measure / Distance Tool | MeasureDistance |
| Tools / Measure / Perimeter Tool | MeasurePerimeter |
| Tools / Measure / Area Tool | MeasureArea |
| Tools / JavaScript / JavaScript Console | JavaScriptConsole |
| Tools / JavaScript / Set Document Actions | SetDocumentActions |
| Tools / JavaScript / Document JavaScript | DocumentJavaScript |

## View tab

| Action | Menu name |
|---|---|
| Zoom / Zoom In | ZoomIn |
| Zoom / Zoom Out | ZoomOut |
| Zoom / Loupe View | LoupeView |
| Zoom / Dynamic Zoom | DynamicZoom |
| *Zoom / Zoom To | ZoomTo |
| Page View / Scroll Options / Single Page | SinglePage |
| Page View / Scroll Options / Continuous | Continuous |
| Page View / Scroll Options / Facing | Facing |
| Page View / Scroll Options / Continuous Facing | ContinuousFacing |
| Page View / Page Fit Options / Actual Size | ActualSize |
| Page View / Page Fit Options / Fit Page | FitPage |
| Page View / Page Fit Options / Fit Width | FitWidth |
| Page View / Page Fit Options / Fit Visible | FitVisible |
| Page View / Page Fit Options / Fit Height | FitHeight |
| Page View / Full Screen | FullScreen |
| Page View / Rotate View / Clockwise | ClockwiseRotateView |
| Page View / Rotate View / Counterclockwise | CounterclockwiseRotateView |
| Page View / Previous View | PreviousView |
| Page View / Next View | NextView |
| New / New Window | NewWindow |
| Display Theme / Change Skin / Purple | ChangeSkinPurple |
| Display Theme / Change Skin / Dark Gray | ChangeSkinGrayDark |
| Display Theme / Change Skin / Light Gray | ChangeSkinGrayLight |
| Display Theme / Change Skin / Blue | ChangeSkinBlue |

| Action | Menu name |
|---|---|
| Current Window / Horizontal Split | HorizontalSplit |
| Current Window / Vertical Split | VerticalSplit |
| Current Window / Quad Split | QuadSplit |
| All Windows / Cascade | Cascade |
| All Windows / Tile / Horizontally | TileHorizontally |
| All Windows / Tile / Vertically | TileVertically |
| All Windows / Close All | CloseAll |

## Comments tab

| Action | Menu name |
|---|---|
| Annotate / Note | Note |
| Annotate / Text Box | TextBox |
| Annotate / Callout | CallOut |
| Annotate / Check Spelling / Check Spelling | CheckSpelling |
| Annotate / Check Spelling / Edit Dictionary | EditDictionary |
| Markup / Highlight / Highlight | Highlight |
| Markup / Highlight / Highlight Area | HighlightArea |
| Markup / Cross-out | Crossout |
| Markup / Underline | Underline |
| Markup / Draw Tools / Line | Line |
| Markup / Draw Tools / Arrow | Arrow |
| Markup / Draw Tools / Rectangle | Rectangle |
| Markup / Draw Tools / Pencil | Pencil |
| Markup / Draw Tools / Oval | Oval |
| Markup / Draw Tools / Polygon | Polygon |
| Markup / Draw Tools / Polygon Line | PolygonLine |
| Markup / Draw Tools / Cloud | Cloud |
| Markup / Draw Tools / Eraser | Eraser |
| Markup / Draw Tools / Hammer | Hammer |
| Markup / Search and Markup | SearchAndMarkup |
| Comment Processing / Import Comments | ImportComments |
| Comment Processing / Export Comments | ExportComments |
| Comment Processing / Migrate Comments | MigrateComments |

| Action | Menu name |
|---|---|
| Comment Processing / Create Comment Summary | CommentSummary |
| Comment Processing / Search Comments | SearchComments |
| Attachments / File | AttachFile |
| Attachments / Audio | AttachSound |

## Advanced Processing tab

| Action | Menu name |
|---|---|
| Process / Flatten File | Flatten |
| Process / Index / Create Full Text Indexes | CreateFullTextIndexes |
| Process / Index / Embedded Index | EmbeddedIndex |
| Process / Reduce / Optimize | Optimize |
| Process / Reduce / Reduce Current File | ReduceCurrentFile |
| Process / Reduce / Reduce Multiple Files | ReduceMultipleFiles |
| Tools / Split | Split |
| Process / Compare | Compare<br><br>Note: There is a hidden AutoCompare control available for automation purposes. See Example 6 for details. |
| File / Previous | PreviousFile |
| File / Next | NextFile |
| File / Scan Inbox | ScanInbox |
| Batch / Convert Assistant | Converter |
| Batch / Sequencer | Workflow |
| Batch / Batch Converter | BatchConvert |
| Batch / Watched Folder | WatchedFolder |
| Read Aloud / Current Page | ReadPage |
| Read Aloud / To End | ReadDoc |
| Read Aloud / To File | ReadToFile |
| Export / Export Area | ExportArea |

## Security tab

| Action | Menu name |
|---|---|
| Redaction / Mark Redaction | MarkRedaction |
| Redaction / Apply Redaction | ApplyReadaction |

| Action | Menu name |
|---|---|
| Redaction / Search and Redact | SearchAndRedact |
| Redaction / Redaction Properties | RedactionToolProperties |
| Sign and Certify / Sign & Mark | FillSign |
| Sign and Certify / Sign / Sign Document | SignDocument |
| Sign and Certify / Time Stamp | TimeStamp |
| Sign and Certify / Certify / Certify Document | CertifyDocument |
| Sign and Certify / Certify / Certify Document Invisibly | CertifyDocumentInvisibly |
| Sign and Certify / Document Signatures / Verify All Signatures | VerifyAllSignatures |
| Sign and Certify / Document Signatures / Clear All Signature Fields | ClearAllSignatureFields |
| Sign and Certify / Document Signatures / View Signed Version | ViewSignedVersion |
| Sign and Certify / Document Signatures / Compare Signed Version to Current Version | CompareSignedVersion |
| Sign and Certify / DocuSign / Send via Docusign | DocuSignSend |
| Sign and Certify / DocuSign / Sign with DocuSign | DocuSignSign |
| Sign and Certify / DocuSign / Forget User | DocuSignForgetUser |
| Sign and Certify / SignDoc / Send via Kofax SignDoc | SignDocSend |
| Sign and Certify / SignDoc / Sign with Kofax SignDoc - Signature | SignDocSign |
| Sign and Certify / SignDoc / Sign with Kofax SignDoc - Image | SignDocImage |
| Security / Clean Document | CleanDocument |
| Security / Remove Elements | RemoveElements |
| Security / Secure Delivery | SecureDelivery |
| Security / Manage Security / Modify | ModifySettings |
| Security / Manage Security / Remove | RemoveSettings |
| IDs and Certificates / Manage Digital Identities | ManageDigitalIDs |
| IDs and Certificates / Trusted Identities | TrustedIDs |
| DRM / RMS Security | RMSSecurity |

## Forms tab

| Action | Menu name |
|---|---|
| Fillable Form / FormTyper | FormTyper |
| Form Elements / Button | Button |
| Form Elements / Radio Button | RadioButton |
| Form Elements / Check Box | CheckBox |

| Action | Menu name |
|---|---|
| Form Elements / Text Field | TextField |
| Form Elements / List Box | ListBox |
| Form Elements / Combo Box | ComboBox |
| Form Elements / Signature Field | DigitalSignature |
| Fields / Set Field Calculation Order | SetFieldCalculationOrder |
| Fields / Highlight Form Fields | HighlightFormFields |
| Fields / Reset Form Fields | ResetFormFields |
| Fields / Set Tab Order | SetTabOrder |
| Data / Import Form Data | ImportFormData |
| Data / Export Form Data | ExportFormData |
| Data / Export Data From Multiple Forms | ExportDataFromMultipleForms |

## Help tab

| Action | Menu name |
|---|---|
| Help / Power PDF Help / Help Online | Help |
| Help / Web Resources / Product Information | ProductInfo |
| Help / Web Resources / Online Shop | OnlineStore |
| Help / Web Resources / Product Support | ProductSupport |
| Help / Product Update | ProductUpdate |
| Help / Automatic Updates | AutomaticUpdates |
| Help / About | About |

## Panels

| Action | Menu name |
|---|---|
| Bookmarks | Bookmarks |
| Pages | PageThumbnails |
| Destinations | Destinations |
| Comments | Comment |
| Attachments | Attachments |
| Stamps | Stamps |
| Watermarks | Watermarks |
| Clip Art | ClipArt |
| Tag | Tag |

| Action | Menu name |
|---|---|
| Compliance | Compliance |
| Form Controls | FormControls |
| Layers | Layers |
| Model Tree | ModelTree |
| Envelope | Envelope |
| Signatures | Signatures |
| Security | Security |
| Sign / Certify | SignAndCertify |
| Send via SignDoc | SignDocSend |
| Collaboration | Collaboration |
| Auto Recovery | AutoRecovery |
| Autostore | AutoStore |
| Reading Order | ReadingOrder |