Table of Contents
PDF Converter Professional 5.0 Automation interface ..................................... 1
Table of Contents ......................................................................................... 2
Interfaces – Overview .................................................................................. 5
IApp interface .................................................................................... 6
   Method CloseAllDocs .......................................................... 6
   Method GetActiveDoc .......................................................... 6
   Method GetActiveTool .......................................................... 7
   Method GetDVDoc .............................................................. 7
   Method GetLanguage ............................................................ 7
   Method GetNumDVDocs ........................................................ 7
   Method Hide ............................................................................. 8
   Method Show ............................................................................. 8
   Method Minimize ...................................................................... 8
   Method Maximize ...................................................................... 8
   Method Restore .......................................................................... 8
   Method SetActiveTool ............................................................. 9
   Method MenuItemExecute ......................................................... 9
   Method MenuItemIsEnabled ....................................................... 9
   Method MenuItemIsMarked ....................................................... 9
   Method MenuItemRemove ......................................................... 10
   Method GetFrame ....................................................................... 10
   Method setFrame ....................................................................... 10
   Method ToolButtonIsEnabled .................................................... 10
   Method ToolButtonRemove ....................................................... 11
IDDDoc interface ........................................................................ 12
   Method Create .......................................................................... 12
   Method Open ............................................................................. 12
   Method Save .............................................................................. 13
   Method Close ............................................................................. 13
   Method DeletePages .................................................................... 13
   Method GetFileName ................................................................... 14
   Method MovePage ....................................................................... 14
   Method ReplacePages ............................................................... 14
   Method InsertPages .................................................................... 15
   Method GetFlags ....................................................................... 15
   Method SetFlags ....................................................................... 16
   Method ClearFlags .................................................................... 16
   Method GetPageMode ............................................................... 16
   Method SetPageMode ............................................................... 17
   Method OpenDVDoc .................................................................... 17
   Method DeleteThumbs ................................................................... 17
   Method CreateThumbs ............................................................... 17
   Method GetInfo .......................................................................... 18
   Method SetInfo .......................................................................... 18
   Method GetPermanentID ............................................................ 18
   Method GetInstanceID ............................................................... 18
   Method CropPages ...................................................................... 19
Method AcquirePage.......................................................................................... 19
Method CreateTextSelect.................................................................................. 19

**IDVDoc interface** .......................................................................................... 20
Method GetDDDoc .............................................................................................. 20
Method Open ......................................................................................................... 20
Method Close ......................................................................................................... 20
Method PrintPages .............................................................................................. 21
Method PrintPagesSilent ..................................................................................... 21
Method GetTitle .................................................................................................... 21
Method Maximize .................................................................................................. 22
Method GetFrame .................................................................................................. 22
Method IsValid ...................................................................................................... 22
Method GetViewMode ........................................................................................... 22
Method SetViewMode ............................................................................................ 23
Method GetDVPageView ....................................................................................... 23
Method SetTextSelection ..................................................................................... 23
Method ShowTextSelect ....................................................................................... 23
Method ClearSelection ......................................................................................... 24
Method BringToFront .............................................................................................. 24
Method SetTitle ..................................................................................................... 24
Method PrintPagesEx ............................................................................................ 25
Method PrintPagesSilentEx ................................................................................. 26

**IDDPage Interface** .......................................................................................... 27
Method GetRotate .................................................................................................. 27
Method SetRotate ................................................................................................... 27
Method GetNumAnnots ......................................................................................... 27
Method GetNumber ............................................................................................... 28
Method CropPage ................................................................................................... 28
Method GetAnnot .................................................................................................... 28
Method AddAnnot .................................................................................................. 29
Method AddNewAnnot ......................................................................................... 29
Method GetDoc ......................................................................................................... 29
Method GetAnnotIndex .......................................................................................... 30
Method GetSize ....................................................................................................... 30
Method CreateWordHilite .................................................................................... 30
Method CreatePageHilite ...................................................................................... 30

**IDDAnnot interface** ....................................................................................... 31
Method GetColor .................................................................................................. 31
Method SetColor .................................................................................................... 31
Method GetDate ...................................................................................................... 31
Method SetDate ....................................................................................................... 32
Method GetRect ....................................................................................................... 32
Method SetRect ....................................................................................................... 32
Method GetTitle ..................................................................................................... 32
Method SetTitle ..................................................................................................... 33
Method GetSubType ............................................................................................... 33
Method IsValid ...................................................................................................... 33
Method isOpen ......................................................................................................... 33
Method SetOpen ...................................................................................................... 34
Method GetContents ............................................................................................... 34
Examples

Example 1 – Inserting pages into a document
Example 2 – Using the MenuItemExecute method of the IApp interface
Example 3 – Highlighting words in a PDF document
Example 4 – Adding annotations

Appendix

The menu names
**Interfaces – Overview**

PDF Converter Professional provides the automation interfaces below. These interfaces provide a wide range of functionality to control the application itself, manage and manipulate PDF documents, their properties, pages and annotations.

<table>
<thead>
<tr>
<th>Interface</th>
<th>Creatable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IApp</td>
<td>Creatable</td>
<td>Controls the PDF Professional application</td>
</tr>
<tr>
<td>IDDDoc</td>
<td>Creatable</td>
<td>Controls the PDF documents</td>
</tr>
<tr>
<td>IDVDoc</td>
<td>Creatable</td>
<td>Controls the view of PDF documents</td>
</tr>
<tr>
<td>IDDPage</td>
<td>Non-creatable</td>
<td>Accessible through IDDDoc interface. Controls the pages within the given PDF document</td>
</tr>
<tr>
<td>IDDAnnot</td>
<td>Non-creatable</td>
<td>Accessible through IDDPage interface</td>
</tr>
<tr>
<td>IDVPageView</td>
<td>Non-creatable</td>
<td>Accessible through IDVDoc interface. Controls viewing PDF pages</td>
</tr>
<tr>
<td>IHiliteList</td>
<td>Creatable</td>
<td>This collection is designed to specify the location of highlighted contents within a PDF document.</td>
</tr>
<tr>
<td>IDDTextSelect</td>
<td>Creatable</td>
<td>This interface is designed to set and show the highlights within a PDF document.</td>
</tr>
<tr>
<td>IRect</td>
<td>Creatable</td>
<td>These interfaces are designed to set or retrieve special properties of different objects</td>
</tr>
<tr>
<td>ITime</td>
<td>Creatable</td>
<td></td>
</tr>
<tr>
<td>IPoint</td>
<td>Creatable</td>
<td></td>
</tr>
</tbody>
</table>

This documentation provides full details of these interfaces and also gives examples for their usage. It also provides the implementation details and relationships between the different interfaces.

Each section provides the definition of a particular method with its input and return parameters, a short description of its usage and points to the related topics.

The documentation also provides several examples. These samples are collected in a separate chapter at the end of the document. The sections of methods contain references to the samples where the particular methods are being used.
**IApp interface**

The IApp interface is designed to drive the PDF Converter Professional application. It provides a control over the application main window and its UI elements, the menus, toolbars and the tools. The opened documents are also accessible through the interface.

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

<table>
<thead>
<tr>
<th>Method Exit</th>
<th>Syntax:</th>
<th>VARIANT_BOOL Exit();</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters:</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>Return value:</td>
<td>VARIANT_TRUE, if pdfplus.exe exited properly.</td>
<td></td>
</tr>
<tr>
<td>Remarks:</td>
<td>Closes all opened documents and quits from the application.</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td>Example 1; Example 2;</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method CloseAllDocs</th>
<th>Syntax:</th>
<th>VARIANT_BOOL CloseAllDocs();</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters:</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>Return value:</td>
<td>Closes all documents opened in the application.</td>
<td></td>
</tr>
<tr>
<td>Remarks:</td>
<td>Closes all documents referred by the IDVDoc interface pointer. The DDDoc objects have to be closed by the Automation client application.</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td>Example 2</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method GetActiveDoc</th>
<th>Syntax:</th>
<th>IDispatch* GetActiveDoc();</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters:</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>Return value:</td>
<td>Retrieves an interface to the active DVDoc object.</td>
<td></td>
</tr>
<tr>
<td>Remarks:</td>
<td>Use it to retrieve the topmost document selected by the user.</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Method GetActiveTool

**Syntax:**

```c++
BSTR GetActiveTool();
```

**Parameters:**

None

**Return value:**

Returns with the name of the selected tool. The return value can be a NULL pointer.

**Remarks:**

Use it to retrieve the name of the selected tool.

**Example:**


### Method GetDVDoc

**Syntax:**

```c++
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.

**Example:**


### Method GetLanguage

**Syntax:**

```c++
BSTR GetLanguage();
```

**Parameters:**

None

**Return value:**

Returns the abbreviated string of the language. The available language codes are the following:

- ENU - English
- DEU - German
- FRA - French
- DUT - Dutch
- ITA - Italian
- ESP - Spanish
- SVE - Swedish
- DAN - Danish

**Remarks:**

Use this method to identify the product installation language.

**Example:**


### Method GetNumDVDocs

**Syntax:**

```c++
long GetNumDVDocs();
```

**Parameters:**

None

**Return value:**

Returns the number of opened documents.

**Remarks:**

You can use this method for enumeration of open documents.

**Example:**
### Method Hide

**Syntax:**

```
VARIANT_BOOL Hide();
```

**Parameters:**

None

**Return value:**

VARIANT_TRUE if the operation succeeded.

**Remarks:**

Hides the application main window.

**Example:**

```vbnet
Dim oApp As IApp
Set oApp = GetObject("{guid}","")
oApp.Hide()
```

### Method Show

**Syntax:**

```
VARIANT_BOOL Show();
```

**Parameters:**

None

**Return value:**

VARIANT_TRUE if the operation succeeded.

**Remarks:**

Makes the application main window visible.

**Example:**

```vbnet
Dim oApp As IApp
Set oApp = GetObject("{guid}","")
oApp.Show()
```

### Method Minimize

**Syntax:**

```
VARIANT_BOOL Minimize(long bMinimize); 
```

**Parameters:**

None

**Return value:**

VARIANT_TRUE if the operation succeeded.

**Remarks:**

Shows the application main window minimized.

**Example:**

```vbnet
Dim oApp As IApp
Set oApp = GetObject("{guid}","")
oApp.Minimize(1)
```

### Method Maximize

**Syntax:**

```
VARIANT_BOOL Maximize(long bMaximize); 
```

**Parameters:**

None

**Return value:**

VARIANT_TRUE if the operation succeeded.

**Remarks:**

Shows the application main window maximized.

**Example:**

```vbnet
Dim oApp As IApp
Set oApp = GetObject("{guid}","")
oApp.Maximize(1)
```

### Method Restore

**Syntax:**

```
VARIANT_BOOL Restore(long bRestore); 
```

**Parameters:**

None

**Return value:**

VARIANT_TRUE if the operation succeeded.

**Remarks:**

Shows the application main window maximized.

**Example:**

```vbnet
Dim oApp As IApp
Set oApp = GetObject("{guid}","")
oApp.Restore(1)
```
### Method SetActiveTool

**Syntax:**

```plaintext
VARIANT_BOOL SetActiveTool(BSTR szButtonName, long bPersistent);
```

**Parameters:**
- **BSTR szButtonName:** The name of the tool.
- **long bPersistent:** if a positive number, the tool will remain active after it has been used, 0 otherwise.

**Return value:**

VARIANT_TRUE if the operation succeeded.

**Remarks:**

Use this method to activate a particular tool.

**Example:**

- Example 2

### Method MenuItemExecute

**Syntax:**

```plaintext
VARIANT_BOOL MenuItemExecute(BSTR szMenuItemName);
```

**Parameters:**
- **BSTR szMenuItemName:** Specifies the action to be performed.

**Return value:**

VARIANT_TRUE if the operation succeeded.

**Remarks:**

Use this method to perform a specific action associated with the given menu item. The possible menu item names are listed in the Appendix.

**Example:**

- Example 2

### Method MenuItemIsEnabled

**Syntax:**

```plaintext
VARIANT_BOOL MenuItemIsEnabled(BSTR szMenuItemName);
```

**Parameters:**
- **BSTR szMenuItemName:** Specifies the menu item.

**Return value:**

Returns VARIANT_TRUE if the menu item is enabled.

**Remarks:**

Before performing the MenuItemExecute method, it might be necessary to check the state of the menu item.

**Example:**

- Example 2

### Method MenuItemIsMarked

**Syntax:**

```plaintext
VARIANT_BOOL MenuItemIsMarked(BSTR szMenuItemName);
```

**Parameters:**
- **BSTR szMenuItemName:** Specifies the menu item.

**Return value:**

Returns VARIANT_TRUE if the menu item is checked.

**Remarks:**

Certain menu items have a ‘checked’ state. You can retrieve this state of a menu item. Not applicable for all menu items.

**Example:**
Method MenuItemRemove

**Syntax:**

```c
VARIANT_BOOL MenuItemRemove(BSTR szMenuItemName);
```

**Parameters:**

- `BSTR szMenuItemName`: Specifies the menu item.

**Return value:**

`VARIANT_TRUE` if the operation succeeded.

**Remarks:**

Use this method if you don’t want to let a user have access to certain menu items. These items will be removed from the menu structure of the application.

**Example:**


Method GetFrame

**Syntax:**

```c
IDispatch* GetFrame();
```

**Parameters:**

- None

**Return value:**

Return a reference to a `Rect` object.

**Remarks:**

Use this method to retrieve the position and the size of the main application window.

**Example:**


Method SetFrame

**Syntax:**

```c
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:**

Use this method if you want to specify the coordinates of the main application window.

**Example:**


Method ToolButtonIsEnabled

**Syntax:**

```c
VARIANT_BOOL ToolButtonIsEnabled(BSTR szButtonName);
```

**Parameters:**

- `BSTR szButtonName`: Specifies the toolbar button.

**Return value:**

Returns `VARIANT_TRUE` if the toolbar button is enabled.

**Remarks:**

Verifies whether the specified button is enabled or disabled.

**Example:**


Method ToolButtonRemove

| Syntax: | VARIANT_BOOL ToolButtonRemove(BSTR szButtonName); |
| Parameters: | BSTR szButtonName: Specifies the toolbar button. |
| Return value: | VARIANT_TRUE if the operation succeeded. |
| Remarks: | You can hide the specified toolbar button while the application runs through automation. |
| Example: |
IDDDoc interface
This interface controls PDF documents at 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**

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>VARIANT_BOOL Create();</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters:</td>
<td>None</td>
</tr>
<tr>
<td>Return value:</td>
<td>VARIANT_TRUE if the operation succeeded.</td>
</tr>
<tr>
<td>Remarks:</td>
<td>Creates a new PDF document. The file does not contain any pages.</td>
</tr>
<tr>
<td>Example:</td>
<td></td>
</tr>
</tbody>
</table>

**Method Open**

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>VARIANT_BOOL Open(BSTR szFullPath);</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters: BSTR szFullPath:</td>
<td>Specifies the PDF file to be opened.</td>
</tr>
<tr>
<td>Return value:</td>
<td>VARIANT_TRUE if the operation succeeded.</td>
</tr>
<tr>
<td>Remarks:</td>
<td>Use this method to open the specified PDF file. A document opened in this way will not be displayed in the application window. Note that the maximum number of open documents is 50.</td>
</tr>
<tr>
<td>Example:</td>
<td><a href="#">Example 1</a></td>
</tr>
</tbody>
</table>


### Method Save

**Syntax:**

```c
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 will always result 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 method `Open`.

**Return value:**

- `VARIANT_BOOL`: if the saving operation succeeded.

**Remarks:**

Saves the content of the document to the specified file.

**Example:**

Example 1

---

### Method Close

**Syntax:**

```c
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:**

```c
VARIANT_BOOL DeletePages(long nStartPage, long nEndPage);
```

**Parameters:**

- `long nStartPage`: Specifies the first page to be deleted.
- `long nEndPage`: Specifies the last page to be deleted.

**Return value:**

- `VARIANT_TRUE`: if the operation succeeded.

**Remarks:**

Deletes the specified pages from the document.

**Example:**

---
## Method GetFileName

**Syntax:**

```c
BSTR GetFileName();
```

**Parameters:**

None

**Return value:**

Returns with the name of the file referenced by the DDDoc object.

**Remarks:**

Use it to retrieve the name of the opened document.

**Example:**


## Method MovePage

**Syntax:**

```c
VARIANT_BOOL MovePage(long nMoveAfterThisPage, long nPageToMove);
```

**Parameters:**

- `long nMoveAfterThisPage`: Specifies the page number after which the moved page will be 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.

**Example:**


## Method ReplacePages

**Syntax:**

```c
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 will be 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, if you want to merge the annotations of the original and the new pages. In case of value 0 the text annotations will not be 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.

**Example:**

*Example 1* is applicable for this method as well.
### Method InsertPages

**Syntax:**

```c
VARIANT_BOOL InsertPages(long nInsertPageAfter, IDispatch* iDDDocSource, long nStartPage, long nNumPages, long bBookmarks);
```

**Parameters:**

- `long nInsertPageAfter`: Specifies a page in the target document after which the new pages will be inserted. Use a value of -1 to insert pages to the beginning of the document.
- `IDispatch* iDDDocSource`: Specifies the source document where the new pages are being inserted from.
- `long nStartPage`: Specifies the very first page of the source document that has to 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 will be 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 places them after the specified page in the current PDF file.

**Example:**

[Example 1](#)

### Method GetFlags

**Syntax:**

```c
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, The document has been modified.
- `DDDocRequiresFullSave` = 2, Incremental saving would not be enough. Use the `DDSaveFull` flag when document is saved.
- `DDDocIsModified` = 4, The document is modified (e.g. bookmarks or annotations have been added).
- `DDDocDeleteOnClose` = 8, The document is a temporary one. It has to be deleted when the file is closed.
- `DDDocWasRepaired` = 16, 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, The document is linearized.
- `DDDocIsOptimized` = 2048, The document is optimized.

**Remarks:**

Use this method to retrieve the current state and other parameters of the document.

**Example:**

[Example](#)
<table>
<thead>
<tr>
<th>Method SetFlags</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Syntax:</strong></td>
</tr>
<tr>
<td><code>VARIANT_BOOL SetFlags(long nFlags);</code></td>
</tr>
<tr>
<td><strong>Parameters:</strong></td>
</tr>
<tr>
<td>long nFlags: The required value of the document flags.</td>
</tr>
<tr>
<td><strong>Return value:</strong></td>
</tr>
<tr>
<td><code>VARIANT_TRUE</code> if the operation succeeded.</td>
</tr>
<tr>
<td><strong>Remarks:</strong></td>
</tr>
<tr>
<td>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. <code>DDDocNeedsSave</code> and <code>DDDocDeleteOnClose</code> flags can be set and cleared.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method ClearFlags</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Syntax:</strong></td>
</tr>
<tr>
<td><code>VARIANT_BOOL ClearFlags(long nFlags);</code></td>
</tr>
<tr>
<td><strong>Parameters:</strong></td>
</tr>
<tr>
<td>long nFlags: Specify the document flags to be cleared.</td>
</tr>
<tr>
<td><strong>Return value:</strong></td>
</tr>
<tr>
<td><code>VARIANT_TRUE</code> if the operation succeeded.</td>
</tr>
<tr>
<td><strong>Remarks:</strong></td>
</tr>
<tr>
<td>Use this flag to clear document flags. Most of the flags are read-only and cannot be set or cleared. <code>DDDocNeedsSave</code> and <code>DDDocDeleteOnClose</code> flags can be set and cleared.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method GetPageMode</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Syntax:</strong></td>
</tr>
<tr>
<td><code>long GetPageMode();</code></td>
</tr>
<tr>
<td><strong>Parameters:</strong></td>
</tr>
<tr>
<td>None</td>
</tr>
<tr>
<td><strong>Return value:</strong></td>
</tr>
<tr>
<td>The return value can be one the following: <code>DDDontCare = 0</code>, <code>DDUseNone = 1</code>, <code>DDUseThumbs = 2</code>, <code>DDUseBookmarks = 3</code>, <code>DFullScreen = 4</code></td>
</tr>
<tr>
<td><strong>Remarks:</strong></td>
</tr>
<tr>
<td>Use this method to retrieve 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.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
</tr>
</tbody>
</table>
## 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.

**Example:**

## 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 will be displayed that shows the document referred by the DDDoc object. This call makes the application visible if it was hidden originally.

**Example:**

## 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 will be removed from the file.

**Return value:**
- `VARIANT_TRUE` if the operation succeeded.

**Remarks:**
Remove embedded thumbnail images of the pages.

**Example:**

## Method CreateThumbs

**Syntax:**
```
VARIANT_BOOL CreateThumbs(long nStartPage, long nEndPage);
```

**Parameters:**
- `long nStartPage`, `nEndPage`: These two values specify the interval of pages. The thumbnails of these pages will be added to the file.

**Return value:**
- `VARIANT_TRUE` if the operation succeeded.

**Remarks:**
Create embedded thumbnail images of the pages or a range of pages.

**Example:**
### 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:**

Use this method to retrieve the document information values like “Author”, “Title”, etc.

**Example:**


### Method SetInfo

**Syntax:**

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

**Parameters:**

- `BSTR szInfoKey`: Specifies the key whose value has to be set.
- `BSTR szBuffer`: Specifies the value.

**Return value:**

`VARIANT_TRUE` if the operation succeeded.

**Remarks:**

Use this method to set the different document information parameters, like “Author”, “Title”, etc.

**Example:**


### Method GetPermanentID

**Syntax:**

```
BSTR GetPermanentID();
```

**Parameters:**

None

**Return value:**

Returns the permanent ID of the document.

**Remarks:**


### Method GetInstanceID

**Syntax:**

```
BSTR GetInstanceID();
```

**Parameters:**

None

**Return value:**

Returns the instance ID of the document.

**Remarks:**


### Method CropPages

**Syntax:**

```c
VARIANT_BOOL CropPages(long nStartPage, long nEndPage, short nEvenOrOddPagesOnly, IDispatch* iRect);
```

**Parameters:**

- `long nStartPage`: Specifies the first page to be cropped.
- `long nEndPage`: Specifies the last page to be cropped.
- `short nEvenOrOddPagesOnly`:
  - `0`: Crop all pages in the range
  - `1`: Crop odd pages in the range
  - `2`: Crop even pages in the range
- `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.

**Example:**

```c
// Example
```

### Method AcquirePage

**Syntax:**

```c
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:**

Use this method to retrieve 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:**

```c
// Example 3; Example 4
```

### Method CreateTextSelect

**Syntax:**

```c
IDispatch* CreateTextSelect(long nPage, IDispatch* iRect);
```

**Parameters:**

- `long nPage`: Specifies the page where the selection should be created.
- `IDispatch* iRect`: Specifies the area of the text selection.

**Return value:**

Returns a reference to a text selection object.

**Remarks:**

Use this method to create a text selection based on a specific area of a page.

**Example:**

```c
// Example
```
**IDVDoc interface**

This interface is strongly related to the DDDoc interface. While DDDoc provides the file-level operations, this interface supplies methods to support viewing documents. Similarly 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 that gives access to file level operations on the document opened in the document window.

### Method GetDDDoc

**Syntax:**

```c
IDispatch* GetDDDoc()
```

**Parameters:**

None

**Return value:**

Returns the IDispatch interface of the associated DDDoc object.

**Remarks:**

This method is used for getting access to the DDDoc object of the DVDoc object. You may need this method if you want to reach the properties of the DDDoc object or call its methods.

**Example:**

Example 3;

### Method Open

**Syntax:**

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

**Parameters:**

- `BSTR szFullPath`: Specifies the PDF file to be opened.

**Return value:**

Return `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:**

```c
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:**

Return value is `VARIANT_TRUE` if the file was closed properly.

**Remarks:**

This method closes the document along with its window.

**Example:**
### 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.
- **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 may 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.

**Return value:** VARIANT_TRUE if the operation succeeded.

**Remarks:** Displays the Print dialog and prints the specified range of pages.

**Example:**

```vbnet
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.
- **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 may 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.

**Return value:** VARIANT_TRUE if the operation succeeded.

**Remarks:** Prints the specified ranges of pages without displaying the Print dialog box.

**Example:**

```vbnet
Method GetTitle

**Syntax:**

BSTR GetTitle();

**Parameters:** None

**Return value:** Returns the title of the of the window associated with the document.

**Remarks:**

**Example:**

```
## 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:**

Use this method to maximize the size of the document window associated to the DVDoc object.

**Example:**

```c
// Example code
Nuance::IDVDoc::Maximize(500);  // Maximize document window
```

## 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 `CPDFPlusRect` object that contains the coordinates of the document window.

**Example:**

```c
// Example code
IDispatch* rect = DVDoc::GetFrame();
```

## Method IsValid

**Syntax:**

```
VARIANT_BOOL IsValid();
```

**Parameters:**

None

**Return value:**

Returns `VARIANT_TRUE` if the document is a valid PDF document.

**Remarks:**

**Example:**

```c
// Example code
if (DVDoc::IsValid())
{
    // Document is valid
}
```

## Method GetViewMode

**Syntax:**

```
long GetViewMode();
```

**Parameters:**

None

**Return value:**

The return value indicates how the document is displayed.

**Remarks:**

The possible return values can be the following:

- `DDDontCare`: 0 -- leave the view mode as it is.
- `DDUseNone`: 1 -- display without bookmark or thumbnails.
- `DDUseThumbs`: 2 -- display using thumbnails.
- `DDUseBookmarks`: 3 -- display using bookmarks.
- `DDFullScreen`: 4 -- display in full screen mode.

**Example:**

```c
// Example code
long viewMode = DVDoc::GetViewMode();
```
### Method SetViewMode

**Syntax:**

```
VARIANT_BOOL SetViewMode(long nType);
```

**Parameters:**

- `long nType`

**Return value:**

- `VARIANT_TRUE` if the operation succeeded.

**Remarks:**

Use this method to modify the view of the document. The possible values are listed under the `GetViewMode` method.

**Example:**

```java

```

### Method GetDVPageView

**Syntax:**

```
IDispatch* GetDVPageView();
```

**Parameters:**

- `None`

**Return value:**

The method returns the `IDispatch` interface of the `DVPageView` object.

**Remarks:**

Use this method if you need access to methods in the `IDVPageView` interface.

**Example:**

```java

```

### Method SetTextSelection

**Syntax:**

```
VARIANT_BOOL SetTextSelection(IDispatch* iDDTextSelect);
```

**Parameters:**

- `IDispatch* iDDTextSelect`: Reference to an `IDDDTextSelect` object that specifies the text selection in the document.

**Return value:**

- `VARIANT_TRUE` if the operation succeeded.

**Remarks:**

Use this method to set the required text selection.

**Example:**

```java

Example 3;
```

### Method ShowTextSelect

**Syntax:**

```
VARIANT_BOOL ShowTextSelect();
```

**Parameters:**

- `None`

**Return value:**

- `VARIANT_TRUE` if the operation succeeded.

**Remarks:**

Call this method to show the selected text.

**Example:**

```java

Example 3;
```
### Method ClearSelection

**Syntax:**
```c
VARIANT_BOOL ClearSelection();
```

**Parameters:**
None

**Return value:**
VARIANT_TRUE if the operation succeeded.

**Remarks:**
Call this method to clear the text selection.

**Example:**

### Method BringToFront

**Syntax:**
```c
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.

**Example:**

### Method SetTitle

**Syntax:**
```c
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:**
Use this method to change the title of the document window.

**Example:**
### 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:**

Returns `VARIANT_TRUE` if the operation succeeded.

**Remarks:**

Displays the Print dialog box and prints the specified pages.

**Example:**

Displays the Print dialog box and prints the specified pages.
### Method PrintPagesSilentEx

**Syntax:**

```c
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.
- `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:**

Returns `VARIANT_TRUE` if the operation succeeded.

**Remarks:**

Prints the specified pages silently without displaying the Print dialog box.

**Example:**

Prints the specified pages silently.
**IDDPage Interface**

This interface represents a page of a PDF document. Certain page level operations – cropping, 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:**

```c
short GetRotate();
```

**Parameters:**

None

**Return value:**

The method returns the rotation value of the page in degrees. The possible values are listed under method SetRotate.

**Remarks:**

Use this method to determine the rotation degree of the page.

**Example:**

```c
Method SetRotate
```

### Method SetRotate

**Syntax:**

```c
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:**

The method returns `VARIANT_TRUE` if the operation succeeded.

**Remarks:**

This method influences the view of the document page. To make changes permanent, the PDF file has to be saved.

**Example:**

```c
Method GetNumAnnots
```

### Method GetNumAnnots

**Syntax:**

```c
long GetNumAnnots();
```

**Parameters:**

None

**Return value:**

This method returns the number of annotations on the given page.

**Remarks:**

The method is used typically when the annotations have to be enumerated.

**Example:**

```c
Example 4;
```
### Method GetNumber

**Syntax:**
```
long GetNumber();
```

**Parameters:** None

**Return value:** This method retrieves the index of the page. Note that the page indexes start from zero.

**Remarks:** Certain methods expect a page index as input parameter. Use this method to retrieve the appropriate value for the given page.

**Example:**

### Method CropPage

**Syntax:**
```
VARIANT_BOOL CropPage(IDispatch* iRect);
```

**Parameters:**
- `IDispatch* iRect`: The IDispatch interface to a Rect object that determines the size of the crop area of the page.

**Return value:** The method returns `VARIANT_TRUE` if the operation succeeded.

**Remarks:** Use the method to adjust the visible area of the page. The coordinates have to be given in user space.

**Example:**

### 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:** Use this method to retrieve the given page annotation. The properties of the annotation can be accessed through the returned interface pointer.

**Example:**
### Method AddAnnot

**Syntax:**
```
VARIANT_BOOL AddAnnot(long nIndexAddAfter, IDispatch* iDDAnnot);
```

**Parameters:**
- `long nIndexAddAfter`: Specifies the index of the new annotation. The annotation will be 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:**
The method returns `VARIANT_BOOL` equal to `VARIANT_TRUE` if the operation succeeded.

**Remarks:**
Use the method to add 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`.

**Example:**

### Method AddNewAnnot

**Syntax:**
```
IDispatch* AddNewAnnot(long nIndexAddAfter, BSTR szSubType, IDispatch* iRect);
```

**Parameters:**
- `long nIndexAddAfter`: Specifies the index of the new annotation. The annotation will be 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.
- `BSTR szSubType`: Specifies the type of annotation. It supports the “Text” subtype only.
- `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 (e.g. color). The return value is NULL if the operation failed.

**Remarks:**
Use this method to create a new annotation on the given page.

**Example:**
```c
Example 4;
```

### Method GetDoc

**Syntax:**
```
IDispatch* GetDoc();
```

**Parameters:**
None

**Return value:**
This method returns the `IDispatch` interface of the `DDDoc` object associated to the `DDPage` object.

**Remarks:**

**Example:**
**Method GetAnnotIndex**

**Syntax:**
```cpp
long GetAnnotIndex(IDispatch* iDDAnnot);
```

**Parameters:**
- `IDispatch* iDDAnnot`: The IDispatch interface of the DDAnnot object whose index has to be retrieved.

**Return value:**
The method returns the zero-based index of the given annotation.

**Remarks:**

**Example:**

**Method GetSize**

**Syntax:**
```cpp
IDispatch* GetSize();
```

**Parameters:**
- None

**Return value:**
The method returns the IDispatch interface of a Point object.

**Remarks:**
Use this method to retrieve the size of the page measured in points.

**Example:**

**Method CreateWordHilite**

**Syntax:**
```cpp
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:**
The method is used for highlighting a series of words on the given page. The DDHiliteList object may specify several series of words.

**Example:**

**Method CreatePageHilite**

**Syntax:**
```cpp
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:**
The method is used for highlighting 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**
IDDA not interface

This interface provides access to properties of the annotations, like color, date, title, content, coordinates measured in user space, open state, 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 having subtype “Text” are creatable.

Method GetColor

Syntax: long GetColor();
Parameters: None
Return value: The method 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 a light blue.
Remarks: Example:

Method SetColor

Syntax: VARIANT_BOOL SetColor(long nRGBColor);
Parameters: long nRGBColor: Specifies the color of the annotation
Return value: The method returns VARIANT_FALSE if the operation succeeded.
Remarks: Example: Example 4;

Method GetDate

Syntax: IDispatch* GetDate();
Parameters: None
Return value: The method returns an IDispatch interface to a Time object.
Remarks: Use the properties of the Time object to retrieve the date of the annotation.
Example:
### Method SetDate

**Syntax:**  
VARIANT_BOOL SetDate(IDispatch* ITime);

**Parameters:**  
IDispatch* ITime: Specifies the creation time of the annotation.

**Return value:**  
The method returns VARIANT_TRUE if the operation succeeded.

**Remarks:**

**Example:**

---

### Method GetRect

**Syntax:**  
IDispatch* GetRect();

**Parameters:**  
None

**Return value:**  
The method returns the IDispatch interface for the Rect object

**Remarks:**

The method returns the bounding rectangle of the annotation. Use the properties of the Rect object to determine the location of the annotation on the page.

**Example:**

---

### Method SetRect

**Syntax:**  
VARIANT_BOOL SetRect(IDispatch* iRect);

**Parameters:**

**Return value:**  
The method returns VARIANT_TRUE if the operation succeeded.

**Remarks:**

Use this method to set or change the location and size of the annotation.

**Example:**

---

### Method GetTitle

**Syntax:**  
BSTR GetTitle();

**Parameters:**  
None

**Return value:**  
The method returns the title of the annotation.

**Remarks:**

**Example:**
## Method SetTitle

**Syntax:**

```vbnet
VARIANT_BOOL SetTitle(BSTR szTitle);
```

**Parameters:**

- `szTitle`: The title to set for the annotation.

**Return value:**

The method returns `VARIANT_TRUE` if the operation succeeded.

**Remarks:**

Use this method to set or change the title of the annotation.

**Example:**

```vbnet
// Example code
SetTitle(szTitle);  // Set title
```

## Method GetSubType

**Syntax:**

```vbnet
BSTR GetSubtype();
```

**Parameters:**

None

**Return value:**

The method returns the subtype property of the annotation.

**Remarks:**

**Example:**

```vbnet
// Example code
GetSubtype();  // Get subtype
```

## Method IsValid

**Syntax:**

```vbnet
VARIANT_BOOL IsValid();
```

**Parameters:**

None

**Return value:**

The method returns `VARIANT_TRUE` if the annotation is valid.

**Remarks:**

The method verifies the annotation object and checks whether or not it is deleted.

**Example:**

```vbnet
// Example code
IsValid();  // Check annotation validity
```

## Method IsOpen

**Syntax:**

```vbnet
VARIANT_BOOL IsOpen();
```

**Parameters:**

None

**Return value:**

The method returns `VARIANT_TRUE` if the text annotation is open.

**Remarks:**

This method verifies whether or not a text annotation is open.

**Example:**

```vbnet
// Example code
IsOpen();  // Check text annotation openness
```
### Method SetOpen

**Syntax:**

```cpp
VARIANT_BOOL SetOpen(long bIsOpen);
```

**Parameters:**

None

**Return value:**

The method returns `VARIANT_TRUE` if the operation succeeded.

**Remarks:**

Use this method to open a text annotation.

**Example:**

```cpp```

### Method GetContents

**Syntax:**

```cpp
BSTR GetContents();
```

**Parameters:**

None

**Return value:**

The method retrieves the content of an annotation.

**Remarks:**

Example:

```cpp```

### Method SetContents

**Syntax:**

```cpp
VARIANT_BOOL SetContents(BSTR szContents);
```

**Parameters:**

None

**Return value:**

The method returns `VARIANT_TRUE` if the operation succeeded.

**Remarks:**

Use this method to set or modify the content of the annotation.

**Example:**

```cpp```

### Method IsEqual

**Syntax:**

```cpp
VARIANT_BOOL IsEqual(IDispatch* iDDAnnot);
```

**Parameters:**

None

**Return value:**

The method returns `VARIANT_TRUE` if the two compared annotations are equal.

**Remarks:**

The method determines whether or not the current annotation is the same as a specified one.

**Example:**

```cpp```
Method Perform

Syntax: 
VARIANT_BOOL Perform(IDispatch* IDVDoc);

Parameters:

Return value: 
The method returns VARIANT_TRUE if the operation succeeded.

Remarks: 
The method refers to Link type annotations. It performs the action associated with the Link object.

Example:
**IDVPageView**

Similarly 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:**

The method returns `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.

**Example:**

### Method DoGoForward

**Syntax:**

```
VARIANT_BOOL DoGoForward();
```

**Parameters:**

None

**Return value:**

The method returns `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.

**Example:**

### Method GetDVDoc

**Syntax:**

```
IDispatch* GetDVDoc();
```

**Parameters:**

None

**Return value:**

The method returns the `IDispatch` interface of the DVDoc object displayed in the page view.

**Remarks:**

Use this method to retrieve the DVDoc object displayed in the view.

**Example:**
### Method GetDoc

**Syntax:**

```c
IDispatch* GetDoc();
```

**Parameters:**

None

**Return value:**

The method returns the IDispatch interface of the DDDoc object displayed in the page view.

**Remarks:**

Use this method to retrieve the DDDoc object displayed in the view.

**Example:**

### Method GetPage

**Syntax:**

```c
IDispatch* GetPage();
```

**Parameters:**

None

**Return value:**

The method returns the IDispatch interface of the DDPage object referring to the current PDF page displayed in the page view.

**Remarks:**

Use this method to retrieve the DDPage object displayed in the view.

**Example:**

### Method GetPageNum

**Syntax:**

```c
long GetPageNum();
```

**Parameters:**

None

**Return value:**

The method returns the zero-based index of the current PDF page displayed in the page view.

**Remarks:**

Use this method to retrieve the index of the displayed page.

**Example:**

### Method GetZoom

**Syntax:**

```c
long GetZoom();
```

**Parameters:**

None

**Return value:**

The method returns the current zoom factor as a percentage.

**Remarks:**

**Example:**
Method GetZoomType

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

Example:

Method Goto

Syntax: VARIANT_BOOL Goto(long nPage);
Parameters: long nPage: Zero based index of the page
Return value: The method returns VARIANT_TRUE if the operation succeeded.
Remarks: Use this method to make a page specified with its index visible.

Example:

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: The method returns VARIANT_TRUE if the operation succeeded.
Remarks: Use this method to scroll the current page to the given position. The coordinates specify the left-top corner of the visible area in user space.

Example:

Method ZoomTo

Syntax: VARIANT_BOOL ZoomTo(short nType, short nScale);
Parameters: short nType: Specifies the type of the Zoom (see also the method GetZoomType)
short nScale: Specifies the scale of zoom as a percentage.
Return value: The method returns VARIANT_TRUE if the operation succeeded.
Remarks: 

Example:
**Method DevicePointToPage**

**Syntax:**
```csharp
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

**Example:**

**Method PointToDevice**

**Syntax:**
```csharp
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.

**Example:**

**Method GetAperture**

**Syntax:**
```csharp
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.

**Example:**
**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.
- short nLength: Specifies the number of objects to be highlighted.

**Return value:**
The method returns 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 case of the method CreateWordHilite.

**Example:**
Example 3;

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

### Method Destroy
**Syntax:**
VARIANT_BOOL Destroy();

**Parameters:**
None

**Return value:**
The method returns VARIANT_TRUE if the operation succeeded.

**Remarks:**
Use this method to delete the current text selection.

**Example:**

### Method GetBoundingRect
**Syntax:**
IDispatch* GetBoundingRect();

**Parameters:**
None

**Return value:**
The method returns a Rect object.

**Remarks:**
The return value of the method provides the bounding rectangle of the current text selection.

**Example:**
Method GetNumText
Syntax: long GetNumText();
Parameters: None
Return value: The method returns the number of selected text objects.
Remarks: 
Example: 

Method GetPage
Syntax: long GetPage();
Parameters: None
Return value: The method returns the index of the page where the selection is being placed.
Remarks: 
Example: 

Method GetText
Syntax: BSTR GetText(long nTextIndex);
Parameters: long nTextIndex: Specifies the index of the selected text.
Return value: The method returns the selected text.
Remarks: Use this method to retrieve the text specified with the given index in the highlighted content.
Example: 

41/51  2/29/2008
**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 also [Example 4](#).

**ITime interface**
This interface defines a Time object having 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 having 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.
Examples
These examples are designed in Visual C++ code. The examples are easily applicable to other programming languages supporting OLE automation.

Example 1 – Inserting pages into a document
The following example demonstrates how pages can be inserted from one PDF document into another one. The sample can also be applied to the methods ReplacePages and Move Pages of the IDDDoc interface

The referenced methods
Creating Application interface
Creating DDDoc interface
 IDDDoc::Open
 IDDDoc::InsertPages
 IDDDoc::Save
 IDDDoc::Close
 IApp::Exit

```cpp
CPDFPlusApp PDFApp;
if( !PDFApp.CreateDispatch(__uuidof(App)) )
{
    return E_FAIL;
}

CPDFPlusDDDoc ddDocTarget;
ddDocTarget.CreateDispatch( __uuidof( DDDoc ) );
if( 0 == ddDocTarget.Open(_T("mydoc.pdf"))
{
    PDFApp.Exit();
    return E_FAIL;
}

CPDFPlusDDDoc ddDocSource;
ddDocSource.CreateDispatch( __uuidof( DDDoc ) );
if( 0 == ddDocSource.Open(_T("pdfpages.pdf"))
{
    ddDocTarget.Close();
    PDFApp.Exit();
    return E_FAIL;
}

// Insert the first two pages of the source document into the 8th position
// of the target document
if( 0 != ddDocTarget.InsertPages( 8, ddDocSource, 0, 2, FALSE ) )
{
    ddDocTarget.Close();
    ddDocSource.Close();
    PDFApp.Exit();
    return E_FAIL;
}

// Save the modifications using DDDoc object
if( 0 == ddDocTarget.Save( DDSaveFull, "mydoc.pdf" ) )
{
```
Example 2 – Using the MenuItemSelected method of the IApp interface

The following example opens a PDF document and creates dynamic form controls on it using the Auto-Form Typer feature.

The referred methods
Creating Application interface
Creating DVDoc interface
IDVDoc::Open
IApp::MenuItemExecute
IApp::MenuItemIsEnabled
IApp::CloseAllDocs
IApp::Exit

CPDFPlusApp PDFApp;
CPDFPlusDVDoc dvOpenDoc;
if( !PDFApp.CreateDispatch(__uuidof(App)) )
    return E_FAIL;
dvOpenDoc.CreateDispatch(__uuidof(DVDoc));
if( dvOpenDoc.Open( CString("mydoc.pdf") ) == 0 )
{
    // File could not be opened
    PDFApp.Exit();
    return E_FAIL;
}
if( !PDFApp.MenuItemIsEnabled( _T("nuance:AutoFormTyper")) )
{
    if( !PDFApp.MenuItemExecute( _T("nuance:AutoFormTyper")) )
    {
        // Failed to perform the action
        PDFApp.CloseAllDocs();
        PDFApp.Exit();
        return E_FAIL;
    }
}
PDFApp.CloseAllDocs();
PDFApp.Exit();

Example 3 – Highlighting words in a PDF document

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

The referred methods
Creating Application interface
Creating DVDoc interface
Creating HiliteList interface
Nuance PDF Converter Professional 5.0 – OLE Automation interface documentation

Examples

IDVDoc::Open
IDDDoc::GetDDDoc
DDDoc::AcquirePage
IHiliteList::Add
DDPage::CreateWordHilite
DVDDoc::SetTextSelection
DVDDoc::ShowTextSelection

CPDFPlusApp PDFApp;
if( !PDFApp.CreateDispatch(__uuidof(App)) )
    return E_FAIL;

//Open the PDF document
CPDFPlusDVDoc dvOpenDoc;
dvOpenDoc.CreateDispatch(__uuidof(DVDoc));
if( dvOpenDoc.Open( "mytest.pdf" ) == 0 )
{
    //"ERROR: can't open test file"
    PDFApp.Exit();
    return E_FAIL;
}

//Get the associated DDDoc object
LPDISPATCH lpdisp = dvOpenDoc.GetDDDoc();
CPDFPlusDDDoc ddDoc(lpdisp);

//Acquire the page to be highlighted
lpdisp = ddDoc.AcquirePage(0);
CPDFPlusDDPage ddPage(lpdisp);

//Create the list of highlights
CPDFPlusHiliteList hl;
hl.CreateDispatch(__uuidof(HiliteList));
hl.Add(12,1);  //12th word should be highlighted
hl.Add(52,2);  //and 2 additional words from the 52nd position

//Create the word highlights and display them
lpdisp = ddPage.CreateWordHilite(hl);
CPDFPlusDDTextSelect ts(lpdisp);
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
Example 4 – Adding annotations

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

The referred methods

Creating Application interface
Creating DVDoc interface

\texttt{DDPage::AcquirePage}

\texttt{Rect properties}

\texttt{DDPage::GetNumAnnots}

\texttt{DDPage::AddNewAnnot}

\texttt{DDAnnot::SetContents}

\texttt{DDAnnot::SetColor}

```c
long AppendTextAnnotation(CPDFPlusDDDoc ddDocOpen, long pageindex, RECT rc, LPCTSTR strComment)
{
    // Acquiring the page of the document
    LPDISPATCH lpdisp = ddDocOpen.AcquirePage(pageindex);
    if (NULL == lpdisp)
    {
        return ERROR_INVALID_PAGEINDEX;
    }
    CPDFPlusDDPage ddPage(lpdisp);
    CPDFPlusRect rcAnnot;
    if (FALSE == rcAnnot.CreateDispatch(__uuidof(Rect)))
    {
        ddPage.ReleaseDispatch();
        return ERROR_INTERNAL;
    }
    rcAnnot.SetLeft(rc.left);
    rcAnnot.SetTop(rc.top);
    rcAnnot.SetRight(rc.right);
    rcAnnot.SetBottom(rc.bottom);
    long index = GetNumAnnots()-1;
    lpDisp = ddPage.AddNewAnnot(index, _T("Text"), rcAnnot);
    if (NULL == lpdisp)
    {
        ddPage.ReleaseDispatch();
        return ERROR_ADD_ANNOT_FAILED;
    }
    CPDFPlusDDAnnot ddAnnot(lpDisp);
    if (VARIANT_FALSE == ddAnnot.SetContents(strComment))
    {
        ddPage.ReleaseDispatch();
        ddAnnot.ReleaseDispatch();
        return ERROR_SETCONTENT;
    }
    ddAnnot.SetColor(128);
    return 0;
}
```
### Appendix

#### The menu names

These menu names can be used as input parameter for the `MenuItemExecute` method of the `IApp` interface.

<table>
<thead>
<tr>
<th>Menu item</th>
<th>Menu name</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>File menu</strong></td>
<td></td>
</tr>
<tr>
<td>Open</td>
<td>Open</td>
</tr>
<tr>
<td>New/Blank PDF</td>
<td>BlankPdf</td>
</tr>
<tr>
<td>New/Form file</td>
<td>FromFile</td>
</tr>
<tr>
<td>New/Create from multiple files</td>
<td>ConvertMultipleFiles</td>
</tr>
<tr>
<td>New/Combine multiple files</td>
<td>CombineMultipleFiles</td>
</tr>
<tr>
<td>New/Overlay multiple files</td>
<td>OverlayMultipleFiles</td>
</tr>
<tr>
<td>New/From clipboard</td>
<td>FromClipboard</td>
</tr>
<tr>
<td>New Package</td>
<td>PackageMultipleFiles</td>
</tr>
<tr>
<td>Close</td>
<td>Close</td>
</tr>
<tr>
<td>Save</td>
<td>Save</td>
</tr>
<tr>
<td>Save As</td>
<td>SaveAs</td>
</tr>
<tr>
<td>Revert to Last Save</td>
<td>Revert</td>
</tr>
<tr>
<td>Email</td>
<td>Email</td>
</tr>
<tr>
<td>Open Revision</td>
<td>open Revision</td>
</tr>
<tr>
<td>Save Revision</td>
<td>save Revision</td>
</tr>
<tr>
<td>Imposition</td>
<td>ZImposition</td>
</tr>
<tr>
<td>Print</td>
<td>Print</td>
</tr>
<tr>
<td>Print with Comment</td>
<td>PrintComment</td>
</tr>
<tr>
<td>Exit</td>
<td>Quit</td>
</tr>
<tr>
<td><strong>Edit menu</strong></td>
<td></td>
</tr>
<tr>
<td>Undo</td>
<td>Undo</td>
</tr>
<tr>
<td>Redo</td>
<td>Redo</td>
</tr>
<tr>
<td>Cut</td>
<td>Cut</td>
</tr>
<tr>
<td>Copy</td>
<td>Copy</td>
</tr>
<tr>
<td>Paste</td>
<td>Paste</td>
</tr>
<tr>
<td>Delete</td>
<td>Delete</td>
</tr>
<tr>
<td>Select All</td>
<td>SelectAll</td>
</tr>
<tr>
<td>Deselect All</td>
<td>DeselectAll</td>
</tr>
<tr>
<td>Add Bookmark</td>
<td>AddBookmark</td>
</tr>
<tr>
<td>Find</td>
<td>Find</td>
</tr>
<tr>
<td>Search</td>
<td>Search</td>
</tr>
<tr>
<td>Preferences</td>
<td>Preference</td>
</tr>
<tr>
<td><strong>View menu</strong></td>
<td></td>
</tr>
<tr>
<td>Go to/First Page</td>
<td>FirstPage</td>
</tr>
<tr>
<td>Go to/Previous Page</td>
<td>PrevPage</td>
</tr>
<tr>
<td>Go to/Next Page</td>
<td>NextPage</td>
</tr>
<tr>
<td>Go to/Last Page</td>
<td>LastPage</td>
</tr>
<tr>
<td>Go to/Go to Page</td>
<td>GoToPage</td>
</tr>
<tr>
<td>Go to/Previous View</td>
<td>PreviousView</td>
</tr>
<tr>
<td>Go to/Next View</td>
<td>NextView</td>
</tr>
</tbody>
</table>
## OLE Automation interface documentation

### Appendix

<table>
<thead>
<tr>
<th>Menu item</th>
<th>Menu name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zoom/Zoom To</td>
<td>ZoomTo</td>
</tr>
<tr>
<td>Zoom/Zoom In</td>
<td>ZoomIn</td>
</tr>
<tr>
<td>Zoom/Zoom Out</td>
<td>ZoomOut</td>
</tr>
<tr>
<td>Zoom/Actual Size</td>
<td>ActualSize</td>
</tr>
<tr>
<td>Zoom/Fit Page</td>
<td>FitPage</td>
</tr>
<tr>
<td>Zoom/Fit Width</td>
<td>FitWidth</td>
</tr>
<tr>
<td>Zoom/Fit Visible</td>
<td>FitVisible</td>
</tr>
<tr>
<td>Zoom/Fit Height</td>
<td>FitHeight</td>
</tr>
<tr>
<td>Single Page</td>
<td>SinglePage</td>
</tr>
<tr>
<td>Continuous</td>
<td>Continuous</td>
</tr>
<tr>
<td>Continuous Facing</td>
<td>ContinuousFacing</td>
</tr>
<tr>
<td>Facing</td>
<td>Facing</td>
</tr>
<tr>
<td>Rotate Clockwise</td>
<td>RotateClockwise</td>
</tr>
<tr>
<td>Rotate Counterclockwise</td>
<td>RotateCounterclockwise</td>
</tr>
<tr>
<td>Toolbars/Casemap</td>
<td>nuance:CaseMap</td>
</tr>
<tr>
<td>Toolbars/Comment</td>
<td>Comment</td>
</tr>
<tr>
<td>Toolbars/Edit</td>
<td>UndoRedo</td>
</tr>
<tr>
<td>Toolbars/File</td>
<td>File</td>
</tr>
<tr>
<td>Toolbars/Form Tools</td>
<td>Form controls</td>
</tr>
<tr>
<td>Toolbars/Hummingbird</td>
<td>Hummingbird</td>
</tr>
<tr>
<td>Toolbars/Measure</td>
<td>Measure</td>
</tr>
<tr>
<td>Toolbars/Security</td>
<td>SecurityPolicy</td>
</tr>
<tr>
<td>Toolbars/Standard</td>
<td>BasicTools</td>
</tr>
<tr>
<td>Toolbars/View</td>
<td>View</td>
</tr>
<tr>
<td>Toolbars/Worksite</td>
<td>WorkSite</td>
</tr>
<tr>
<td>Toolbars/Zoom</td>
<td>Zoom</td>
</tr>
<tr>
<td>Toolbars/Reset Toolbars</td>
<td>ResetToolBar</td>
</tr>
<tr>
<td>Toolbars/Hide Toolbars</td>
<td>ShowToolBar</td>
</tr>
<tr>
<td>Toolbars/Show Button Labels/Default Labels</td>
<td>DefaultLabel</td>
</tr>
<tr>
<td>Toolbars/Show Button Labels/No Labels</td>
<td>NoLabel</td>
</tr>
<tr>
<td>Toolbars/Show Button Labels/All Labels</td>
<td>AllLabel</td>
</tr>
<tr>
<td>Toolbars/Customize Toolbars</td>
<td>CustomizeToolBar</td>
</tr>
<tr>
<td>Navigation Panels/Attachment</td>
<td>Attachment</td>
</tr>
<tr>
<td>Navigation Panels/Bookmark</td>
<td>Bookmark</td>
</tr>
<tr>
<td>Navigation Panels/Comments</td>
<td>Comments</td>
</tr>
<tr>
<td>Navigation Panels/Destinations</td>
<td>Destinations</td>
</tr>
<tr>
<td>Navigation Panels/Form Controls</td>
<td>nuance:FormTyperPanel</td>
</tr>
<tr>
<td>Navigation Panels/Model Tree</td>
<td>Z3DtreePanel</td>
</tr>
<tr>
<td>Navigation Panels/Pages</td>
<td>Thumbnail</td>
</tr>
<tr>
<td>Navigation Panels/Signatures</td>
<td>DigSigPanel</td>
</tr>
<tr>
<td>Clipart</td>
<td>ClipArt</td>
</tr>
<tr>
<td>Envelope</td>
<td>Delivery</td>
</tr>
<tr>
<td>Security</td>
<td>Security</td>
</tr>
<tr>
<td>Sign/Certify</td>
<td>Signature</td>
</tr>
<tr>
<td>Stamps</td>
<td>Stamps</td>
</tr>
<tr>
<td>Watermarks</td>
<td>Watermark</td>
</tr>
<tr>
<td>Menu Bar</td>
<td>HideMenuBar</td>
</tr>
<tr>
<td>Menu item</td>
<td>Menu name</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>Full Screen</td>
<td>FullScreen</td>
</tr>
<tr>
<td>Grid</td>
<td>Grid</td>
</tr>
<tr>
<td>Snap to Grid</td>
<td>Snap</td>
</tr>
<tr>
<td>Rulers</td>
<td>Rulers</td>
</tr>
<tr>
<td>Guides</td>
<td>Guides</td>
</tr>
</tbody>
</table>

**Document menu**

<table>
<thead>
<tr>
<th>Menu item</th>
<th>Menu name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pages/New Page</td>
<td>NewPage</td>
</tr>
<tr>
<td>Pages/Insert Page</td>
<td>InsertPage</td>
</tr>
<tr>
<td>Pages/Extract Page</td>
<td>ExtractPage</td>
</tr>
<tr>
<td>Pages/Delete Page</td>
<td>DeletePage</td>
</tr>
<tr>
<td>Pages/Crop Page</td>
<td>CropPage</td>
</tr>
<tr>
<td>Pages/Rotate Page</td>
<td>RotatePage</td>
</tr>
<tr>
<td>Watermark/Add</td>
<td>Watermark:AddHeadersFootersWatermarks</td>
</tr>
<tr>
<td>Watermark/Delete</td>
<td>ZeonForm:DelWatermark</td>
</tr>
<tr>
<td>Insert Clip Art</td>
<td>ZEON:InsertClipArt</td>
</tr>
<tr>
<td>Sign &amp; Certify/Sign Document</td>
<td>DigSigSignDoc</td>
</tr>
<tr>
<td>Sign &amp; Certify/Certify Document</td>
<td>CertifyVisible</td>
</tr>
<tr>
<td>Sign &amp; Certify/ Certify Document invisibly</td>
<td>CertifyInvisible</td>
</tr>
<tr>
<td>Sign &amp; Certify/Sign Signature field</td>
<td>SignSignatureField</td>
</tr>
<tr>
<td>Sign &amp; Certify/Sign/Certify Panel</td>
<td>SignCertifyPanel</td>
</tr>
<tr>
<td>Sign &amp; Certify/Verify All Signatures</td>
<td>DigSigVerifyAll</td>
</tr>
<tr>
<td>Sign &amp; Certify/Clear All Signature Fields</td>
<td>ClearAllField</td>
</tr>
<tr>
<td>Sign &amp; Certify/View Signed Version</td>
<td>DigSigViewVersion</td>
</tr>
<tr>
<td>Sign &amp; Certify/Compare Signed Version to Current Version</td>
<td>DigSigCompare</td>
</tr>
<tr>
<td>Security /Secure Document</td>
<td>ShowSecurityPanel</td>
</tr>
<tr>
<td>Security /Security Properties</td>
<td>SecurityProperty</td>
</tr>
<tr>
<td>Security /Remove Security Settings</td>
<td>RemoveSecuritySettings</td>
</tr>
<tr>
<td>Security /Secure Delivery</td>
<td>Delivery</td>
</tr>
<tr>
<td>Trusted Identities</td>
<td>Trusted Identity</td>
</tr>
<tr>
<td>Manage Digital IDs</td>
<td>Security Settings</td>
</tr>
<tr>
<td>Redaction/Mark Redaction</td>
<td>MarkRedaction</td>
</tr>
<tr>
<td>Redaction/Redaction Tool Properties</td>
<td>RedactionToolProperties</td>
</tr>
<tr>
<td>Redaction/Search and Redaction</td>
<td>RedactionSearch</td>
</tr>
<tr>
<td>Redaction/Apply Redaction</td>
<td>RedactionApply</td>
</tr>
<tr>
<td>Inspect Document</td>
<td>ExamineDoc</td>
</tr>
<tr>
<td>Bates Stamping/Add</td>
<td>ZEON:AddBatesStamp</td>
</tr>
<tr>
<td>Bates Stamping/Remove</td>
<td>ZEON:RemoveBatesStamp</td>
</tr>
<tr>
<td>Compare documents</td>
<td>ZEON:CompareDoc</td>
</tr>
<tr>
<td>Split document</td>
<td>Zeon:FileSplit</td>
</tr>
<tr>
<td>JavaScript /JavaScript Console</td>
<td>JavaScript Console</td>
</tr>
<tr>
<td>JavaScript /Document JavaScripts</td>
<td>Document JavaScripts</td>
</tr>
<tr>
<td>JavaScript /Set Document Actions</td>
<td>Set Document Actions</td>
</tr>
<tr>
<td>Create Full Text Indexes</td>
<td>Catalog</td>
</tr>
<tr>
<td>Embedded Index</td>
<td>Embed_Index</td>
</tr>
<tr>
<td>Document Properties</td>
<td>DocumentProperties</td>
</tr>
</tbody>
</table>

**Comments menu**
<table>
<thead>
<tr>
<th>Menu item</th>
<th>Menu name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note</td>
<td>Note</td>
</tr>
<tr>
<td>Text Box</td>
<td>FreeText</td>
</tr>
<tr>
<td>Callout</td>
<td>CallOut</td>
</tr>
<tr>
<td>Text Markups/Text Markup Tool</td>
<td>Markup</td>
</tr>
<tr>
<td>Text Markups/Insert Text at Cursor</td>
<td>Insert</td>
</tr>
<tr>
<td>Text Markups/Replace Selected Text</td>
<td>Replace</td>
</tr>
<tr>
<td>Text Markups/Add Note to Selected Text</td>
<td>NoteText</td>
</tr>
<tr>
<td>Text Markups/Highlight Selected Text</td>
<td>HighlightText</td>
</tr>
<tr>
<td>Text Markups/Underline Selected Text</td>
<td>UnderlineText</td>
</tr>
<tr>
<td>Text Markups/Cross Out Text for Deletion</td>
<td>CrossoutText</td>
</tr>
<tr>
<td>Stamp</td>
<td>Stamp</td>
</tr>
<tr>
<td>Highlight</td>
<td>Highlight</td>
</tr>
<tr>
<td>Cross-out</td>
<td>CrossOut</td>
</tr>
<tr>
<td>Underline</td>
<td>UnderLine</td>
</tr>
<tr>
<td>Draw Tools/Line</td>
<td>Line</td>
</tr>
<tr>
<td>Draw Tools/Rectangle</td>
<td>Rectangle</td>
</tr>
<tr>
<td>Draw Tools/Pencil</td>
<td>Pencil</td>
</tr>
<tr>
<td>Draw Tools/Oval</td>
<td>Oval</td>
</tr>
<tr>
<td>Draw Tools/Polygon</td>
<td>Polygon</td>
</tr>
<tr>
<td>Draw Tools/Polygon Line</td>
<td>PolyLine</td>
</tr>
<tr>
<td>Draw Tools/Cloud</td>
<td>Cloudy</td>
</tr>
<tr>
<td>Draw Tools/Erase</td>
<td>Erase</td>
</tr>
<tr>
<td>Draw Tools/Hammer</td>
<td>Hammer</td>
</tr>
<tr>
<td>Attach a File</td>
<td>AttachFile</td>
</tr>
<tr>
<td>Attach a Sound</td>
<td>AttachSound</td>
</tr>
<tr>
<td>Create Comment Summary</td>
<td>CreateCommentSummary</td>
</tr>
<tr>
<td>Export Comments</td>
<td>ExportComment</td>
</tr>
<tr>
<td>Import Comments</td>
<td>ImportComment</td>
</tr>
<tr>
<td>Migrate Comments</td>
<td>MigrateComment</td>
</tr>
<tr>
<td>Search Comments</td>
<td>SearchComments</td>
</tr>
<tr>
<td>Show Comments Panel</td>
<td>ShowCommentsPanel</td>
</tr>
</tbody>
</table>

### Forms menu

<table>
<thead>
<tr>
<th>Button tool</th>
<th>Form:Button</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check Box Tool</td>
<td>Form:Checkbox</td>
</tr>
<tr>
<td>Radio Button Tool</td>
<td>Form:RadioButton</td>
</tr>
<tr>
<td>Combo Box Tool</td>
<td>Form:Combobox</td>
</tr>
<tr>
<td>List Box Tool</td>
<td>Form:Listbox</td>
</tr>
<tr>
<td>Text Field Tool</td>
<td>Form:Textbox</td>
</tr>
<tr>
<td>Digital Signature Tool</td>
<td>Form:Signature</td>
</tr>
<tr>
<td>Set Tab Order</td>
<td>Form:SetOrder</td>
</tr>
<tr>
<td>Set Field Calculation Order</td>
<td>ZeonForm:CalcOrder</td>
</tr>
<tr>
<td>Highlight Form Fields</td>
<td>Form:Sep</td>
</tr>
<tr>
<td>Reset Form Fields</td>
<td>Form:Highlight</td>
</tr>
<tr>
<td>Import Data to Form</td>
<td>ZeonForm:ImportFDF</td>
</tr>
<tr>
<td>Export Data from Form</td>
<td>ZeonForm:ExportFDF</td>
</tr>
<tr>
<td>Form Typer</td>
<td>nuance:AutoFormTyper</td>
</tr>
<tr>
<td>Menu item</td>
<td>Menu name</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>Standard/ Hand</td>
<td>Hand</td>
</tr>
<tr>
<td>Standard/ Select Text</td>
<td>SelectText</td>
</tr>
<tr>
<td>Standard/ Select Image</td>
<td>SelectImage</td>
</tr>
<tr>
<td>Standard/ Select Area</td>
<td>SelectArea</td>
</tr>
<tr>
<td>Standard/ Select Object</td>
<td>SelectObject</td>
</tr>
<tr>
<td>Standard/ Link</td>
<td>LinkTool</td>
</tr>
<tr>
<td>Standard/Crop</td>
<td>CropTool</td>
</tr>
<tr>
<td>Zoom/Zoom In</td>
<td>ZoomInTool</td>
</tr>
<tr>
<td>Zoom/Zoom Out</td>
<td>ZoomOutTool</td>
</tr>
<tr>
<td>Zoom/ Dynamic Zoom</td>
<td>ZoomDynamic</td>
</tr>
<tr>
<td>Zoom/ Loupe</td>
<td>Louise</td>
</tr>
<tr>
<td>Advanced/Movie Tool</td>
<td>Zeon:Movie</td>
</tr>
<tr>
<td>Advanced/Sound Tool</td>
<td>Zeon:Sound</td>
</tr>
<tr>
<td>Advanced/3D Tool</td>
<td>3DTool</td>
</tr>
<tr>
<td>Measuring/Distance Tool</td>
<td>Zeon:Distance</td>
</tr>
<tr>
<td>Measuring/Perimeter Tool</td>
<td>Zeon:Perimeter</td>
</tr>
<tr>
<td>Measuring/Area Tool</td>
<td>Zeon:Area</td>
</tr>
<tr>
<td>Optimize PDF</td>
<td>nuance:Optimize</td>
</tr>
<tr>
<td>Tag PDF</td>
<td>nuance:Retag</td>
</tr>
<tr>
<td>Make Searchable PDF</td>
<td>&amp;Make Searchable PDF</td>
</tr>
<tr>
<td>Read PDF/Read Current Page</td>
<td>nuance:TTS_PAGE</td>
</tr>
<tr>
<td>Read PDF/Read to End of Document</td>
<td>nuance:TTS_DOC</td>
</tr>
<tr>
<td>Read PDF/Read to File</td>
<td>nuance:TTS_FILE</td>
</tr>
<tr>
<td>Read PDF/Prev. para</td>
<td>nuance:TTS_PREVPARA</td>
</tr>
<tr>
<td>Read PDF/Next Para</td>
<td>nuance:TTS_NEXTPARA</td>
</tr>
<tr>
<td>Read PDF/Prev. Page</td>
<td>nuance:TTS_PREVPAGE</td>
</tr>
<tr>
<td>Read PDF/Next Page</td>
<td>nuance:TTS_NEXTPAGE</td>
</tr>
<tr>
<td>Read PDF/Pause</td>
<td>nuance:TTS_PAUSE</td>
</tr>
<tr>
<td>Read PDF/Stop</td>
<td>nuance:TTS_STOP</td>
</tr>
<tr>
<td>Read PDF/Properties</td>
<td>nuance:TTS_PROP</td>
</tr>
<tr>
<td>Touchup/Touchup Text</td>
<td>TouchUpTextTool</td>
</tr>
<tr>
<td>Touchup/Touchup Object</td>
<td>TouchUpObjectTool</td>
</tr>
</tbody>
</table>

**Favorites menu**

Add To Favorites | AddToFavorites
Manage Favorites | ManageFavorites

**Window menu**

New Window | NewWindow
Cascade | Cascade
Tile/ Horizontally | TileHorizontal
Tile/ Vertically | TileVertical
Close All | CloseAll
Horizontal Split | SplitHorz
Vertical Split | SplitVert
Quad Split | SpreadsheetSplit