



# Reference manual

for Sorax PDF SDK - ActiveX Edition  
version 1.10

## Contents

Features.....	3
<i>Supported PDF version</i> .....	3
<i>Fonts</i> .....	3
<i>Streams</i> .....	3
<i>Colors</i> .....	3
<i>Patterns</i> .....	3
<i>Printing</i> .....	3
<i>Encryption</i> .....	3
Methods.....	4
<i>ResetConfig</i> .....	4
<i>OpenDoc</i> .....	4
<i>CloseDoc</i> .....	4
<i>GetPageCount</i> .....	5
<i>GetDocInfo</i> .....	5
<i>GetDocOutline</i> .....	5
<i>ExportDoc</i> .....	5
<i>PrintDirect</i> .....	6
<i>CreateView</i> .....	6
<i>DestroyView</i> .....	7
<i>AttachDoc</i> .....	7
<i>DetachDoc</i> .....	7
Events.....	9
<i>SelectionChanged</i> .....	9
<i>PageChanged</i> .....	9
<i>DocumentChanged</i> .....	9
<i>InsertOutlineItem</i> .....	10
<i>Searching</i> .....	10
Constants .....	10
<i>for the static arrays of the SPDINFO structure</i> .....	10
<i>Export types</i> .....	10
<i>Error codes</i> .....	11
Initializer file.....	11
<i>General section</i> .....	11
<i>FontMap section</i> .....	11
<i>UnicodeMap section</i> .....	11
<i>Export section</i> .....	12



## Features

### ***Supported PDF version***

PDF 1.6 standard. Handling of linearized and non- linearized PDF files.

### ***Fonts***

Handling of Type1, Type3, TrueType simple and CID, Type0 composite embedded or external fonts.

### ***Streams***

ASCII hexadecimal, ASCII base-85, LZW and Flate, Run length, Group 3 or Group 4 CCITT facsimile (fax), JBIG2, DCT and JPX decode filters

### ***Colors***

Handling of RGB, Gray, CMYK, ICCBased, Lab, Indexed, Separation color formats.

### ***Patterns***

Tiling and shadings patterns

### ***Printing***

PS, EPS, BMP printing. OPI support.

### ***Encryption***

Supporting of 40 and 128 bit encryption-level. Handling of User and Owner password.

## Methods

### ***ResetConfig***

This function sets the initialization file, that necessary to handle PDF files.

**VARIANT\_BOOL ResetConfig(LPSTR lpszFileName);**

#### **Parameters**

*lpszFileName*

Name and path of the initialization file.

#### **Return value**

TRUE, if the initialization file exists and processed, otherwise FALSE.

### ***OpenDoc***

Opens a PDF document.

**OLE\_HANDLE OpenDoc(LPSTR lpszFileName,  
LPSTR lpszUserPwd,  
LPSTR lpszOwnerPwd);**

#### **Parameters**

*lpszFileName*

Name and path of the PDF file, witch is to be opened.

*lpszUserPwd*

User password for opening.

*lpszOwnerPwd*

Owner password.

#### **Return value**

Handle of the document, in case of successful opening. NULL, in case of opening failure, and sets the error code, wich is accessible with GetLastError().

### ***CloseDoc***

Closes an opened PDF document.

**BOOL CloseDoc(OLE\_HANDLE hDoc);**

#### **Parameters**

*hDoc*

Handle of the document.

#### **Return value**

TRUE, in case of successful reading. FALSE, in case of invalid document handle.

## ***GetPageCount***

Gets the number of pages.

**int** GetPageCount(OLE\_HANDLE hDoc);

### **Parameters**

*hDoc*

Handle of the document.

### **Return value**

Number of the pages, or -1, in case of invalid document handle.

## ***GetDocInfo***

Gets the information-structure of a PDF file.

**BSTR** GetDocInfo(OLE\_HANDLE hDoc);

### **Parameters**

*hDoc*

Document handle.

### **Return value**

A character string, separated with '|' characters. The string represents the information-structure.

## ***GetDocOutline***

Processes the bookmarks of a PDF document.

**LONG** GetDocOutline(OLE\_HANDLE hDoc);

### **Parameters**

*hDoc*

Handle of document.

### **Return value**

Number of entries, or -1, in case of invalid document handle.

## ***ExportDoc***

Export pages from PDF document.

**VARIANT\_BOOL** ExportDoc(OLE\_HANDLE hDoc,  
    **BSTR** lpszFileName,  
    **LONG** nFromPage,  
    **LONG** nToPage,  
    **LONG** nExpType);

### **Parameters**

*hDoc*

Handle of document.

*lpzFileName*

Output file name.

*nFromPage*

The first page in the range to be exported.

*nToPage*

The last page in the range to be exported.

*nExpType*

Type of export.

### **Return value**

TRUE, if the processing was successful. FALSE, in case of invalid document handle.

## ***PrintDirect***

Allows to send PDF data directly to the printer.

```
VARIANT_BOOL PrintDirect(OLE_HANDLE hDoc,  
    BSTR lpzPrinterName,  
    LONG nFromPage,  
    LONG nToPage,  
    SHORT nFlags);
```

### **Parameters**

*hDoc*

Handle of document.

*lpzPrinterName*

Name of printer.

*nFromPage*

The first page in the range to be printed.

*nToPage*

The last page in the range to be printed.

*nFlags*

The printing mode and the other settings belonging to the chosen mode can be set here. The possible values of the settings can be found under the entry-word „Constants”.

### **Return value**

TRUE, if the processing was successful. FALSE, in case of invalid document handle.

## ***CreateView***

Creates a PDF-View window.

```
OLE_HANDLE CreateView(OLE_HANDLE hDoc,  
    LONG nLeft, LONG nTop, LONG nRight, LONG nBottom,  
    OLE_HANDLE hParentWnd,  
    UINT uID);
```

### **Parameters**

*hDoc*

Handle of document.

*nLeft, nTop, nRight, nBottom*

Specifies the size and the position of the window.

*hParentWnd*

Handle of parent window.

*nID*

Identifier of the window.

#### **Return value**

Handle of the created window, or NULL, in case of error. Error code is accessible with GetLastError().

### ***DestroyView***

Destroys a PDF-View window.

**void DestroyView(OLE\_HANDLE hWnd);**

#### **Parameters**

*hWnd*

Handle of the window.

### ***AttachDoc***

Attaches a document handle to a window.

**VARIANT\_BOOL AttachDoc(OLE\_HANDLE hWnd,  
OLE\_HANDLE hDoc);**

#### **Parameters**

*hWnd*

Handle of the window.

*hDoc*

Document handle.

#### **Return value**

TRUE, in case of successful processing, otherwise FALSE.

### ***DetachDoc***

Detaches a document handle from a window.

**OLE\_HANDLE DetachDoc(OLE\_HANDLE hWnd);**

#### **Parameters**

*hWnd*

Handle of the window.

#### **Return value**

Handle of the detached document, in case of successful processing, otherwise 0.

## **ShowPage**

Displays a PDF page in the window.

```
void ShowPage(OLE_HANDLE hWnd,  
              LONG nPage,  
              FLOAT fDPI);
```

### **Parameters**

*hWnd*

Handle of the window.

*nPage*

The number of the page to be displayed.

*fDPI*

The scale of resolution (in DPI) of the page to be displayed.

## **RotatePage**

Rotation of a PDF page.

```
void RotatePage(OLE_HANDLE hWnd,  
                LONG nRotate);
```

### **Parameters**

*hWnd*

Handle of the window.

*nRotate*

The value of the measure of rotation. Possible values: 0, 90, 180, 270.

## **Copy**

Copies the selected content of a PDF page to the clipboard.

```
VARIANT_BOOL Copy(OLE_HANDLE hWnd);
```

### **Parameters**

*hWnd*

Handle of the window.

### **Return value**

TRUE, in case of successful processing, otherwise FALSE.

## **Print**

Printing PDF page(s).

```
LONG Print(OLE_HANDLE hWnd);
```

### **Parameters**

*hWnd*

Handle of the window.

### **Return value**

0, in case of successful processing, otherwise -1.

## ***Find***

Searching for text in the document.

**void Find(OLE\_HANDLE hWnd);**

### **Parameters**

*hWnd*

Handle of the window.

## **Events**

### ***SelectionChanged***

The state of the textual selection has changed on the displayed page.

**void SelectionChanged(VARIANT\_BOOL bSelection);**

### **Parameters**

*bSelection*

Indicates, that text has been selected or a selection has been put an end of.

### ***PageChanged***

The displayed page, or its attributes has been changed.

**void PageChanged(LONG nPage,  
FLOAT fDPI,  
VARIANT\_BOOL\* pbShowPage);**

### **Parameters**

*nPage*

Page number.

*fDPI*

Resolution.

*pbShowPage*

Redraw setting.

### ***DocumentChanged***

The displayed document of the window has been changed.

**void DocumentChanged(BSTR lpszFileName);**

### **Parameter**



## **Error codes**

<b>ERROR_INVALID_FILE</b>	<b>7001</b>
File open failure. Invalid file name or incorrect PDF file.	
<b>ERROR_INVALID_USER_PASSWORD</b>	<b>7002</b>
Invalid user password.	
<b>ERROR_INVALID_DOCUMENT_HANDLE</b>	<b>7003</b>
Invalid document handle.	
<b>ERROR_INVALID_VIEW_HANDLE</b>	<b>7004</b>
Invalid view window handle.	
<b>ERROR_PAGE_NOT_EXISTS</b>	<b>7005</b>
Nonexistent page.	
<b>ERROR_PERM_COPY</b>	<b>7006</b>
To copy to the clipboard is not permitted.	
<b>ERROR_PERM_PRINT</b>	<b>7007</b>
Printing is not permitted.	

## **Initializer file**

Specifying the default settings of the PDF view window.

### **General section**

FontPath - Enumeration of the fonts' access paths.  
TextEncoding - Encoding of the text.

### **FontMap section**

Assigning the fonts.

Syntax:

FontName=ExtFontFile, where

FontName - Name of the Font.

ExtFontFile - External font file specified with complete access path.

For example:

Courier=c:\SPdf\resources\fonts\x1.pfb

Courier-Bold= c:\SPdf\resources\fonts\x2.pfb

·  
·  
·

### **UnicodeMap section**

Assigning unicode characters.

Syntax:

UnicodeName=ExtUnicodeFile, where

UnicodeName – Name of codeset .

ExtUnicodeFile – External unicode file specified with complete access path.

For example:

LatinN=c:\SPdf\resources\UnicodeMap\LatinN.pfb

.  
. .  
.

## ***Export section***

Assigning unicode characters.

Bmp.DPI - Resolution of exported images (default: 72).  
Xml.Text - Add text to the exported xml file (default: no).  
Xml.Font - Add font changes to the exported xml file (default: yes).  
Xml.Graph - Add vector graphics to the exported xml file (default: no).  
Xml.EmbeddedImage - Extract embedded images from the pdf source file and save them (default: no).  
Xml.EmbeddedFonts - Extract embedded images from the pdf source file and save them (default: no).

You will be able to find the extracted files in ElementsOf.<pdf\_src> directory, whereabouts the pdf\_src string is the title of PDFsource file.