



Reference manual

for Sorax PDF SDK - DLL 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
Functions	3
<i>SPD_ResetConfig</i>	3
<i>SPD_Open</i>	4
<i>SPD_Close</i>	4
<i>SPD_GetPageCount</i>	4
<i>SPD_GetInfo</i>	4
<i>SPD_GetMetaData</i>	5
<i>SPD_GetOutline</i>	5
<i>SPD_GetPageText</i>	6
<i>SPD_Export</i>	6
<i>SPD_PrintDirect</i>	7
<i>SPV_Create</i>	7
Messages	7
<i>View window</i>	8
<i>Notifications</i>	10
Declarations	12
<i>HSPDFDOC</i>	12
<i>SPDINFO</i>	12
<i>NMSPVACTION</i>	12
<i>SPDOUTLINEPROC</i>	13
<i>SPDEXPORTPROC</i>	13
Constants	13
<i>for the static arrays of the SPDINFO structure</i>	13
<i>For printing</i>	14
<i>Export types</i>	14
<i>Error codes</i>	14
Initializer file.....	15
<i>General section</i>	15
<i>FontMap section</i>	15
<i>UnicodeMap section</i>	15
<i>Export section</i>	15

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.

Functions

SPD_ResetConfig

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

BOOL SPD_ResetConfig(LPSTR lpszFileName);

Parameters

lpszFileName

Name and path of the initialization file.

Return value

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



SPD_Open

Opens a PDF document.

```
HSPDFDOC SPD_Open(LPSTR lpszFileName,  
LPSTR lpszUserPwd,  
LPSTR lpszOwnerPwd);
```

Parameters

lpszFileName

Name and path of the PDF file, which 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, which is accessible with GetLastError().

SPD_Close

Closes an opened PDF document.

```
BOOL SPD_Close(HSPDFDOC hDoc);
```

Parameters

hDoc

Handle of the document.

Return value

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

SPD_GetPageCount

Gets the number of pages.

```
int SPD_GetPageCount(HSPDFDOC hDoc);
```

Parameters

hDoc

Handle of the document.

Return value

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

SPD_GetInfo

Gets the information-structure of a PDF file.

```
BOOL SPD_GetInfo(HSPDFDOC hDoc,  
PXAPDINFO pInfo);
```

Parameters

hDoc
Document handle.

pInfo
Address of the PSPDFINFO structure.

Return value

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

SPD_GetMetaData

Gets the meta datas of a PDF document.

```
int SPD_GetMetaData(HSPDFDOC hDoc,  
                   LPTSTR lpszBuf,  
                   size_t nSize);
```

Parameters

hDoc
Handle of the document.

lpszBuf
Address of the buffer, where the datas should be placed.

nSize
Size of the buffer.

Return value

Length of meta datas, or -1, in case of invalid document handle or data type.

SPD_GetOutline

Processes the bookmarks of a PDF document.

```
int SPD_GetOutline(HSPDFDOC hDoc,  
                  SPDOUTLINEPROC pCb,  
                  LPVOID pvParam);
```

Parameters

hDoc
Handle of document.

pCb
Points to the SPDOUTLINEPROC function.

pvParam
User-defined data.

Return value

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

SPD_GetPageText

Gets the textual contents of a specified page.

```
int SPD_GetPageText(HSPDFDOC hDoc,  
                    int nPage,  
                    LPSTR lpszBuf,  
                    size_t nSize);
```

Parameters

hDoc

Handle of document.

nPage

Page number.

lpszBuf

Address of the buffer, where the data should be placed.

nSize

Size of the buffer.

Return value

Number of characters on the specified page, or -1, in case of invalid document handle.

SPD_Export

Export pages from PDF document.

```
BOOL SPD_Export(HSPDFDOC hDoc,  
                LPTSTR lpszFileName,  
                int nFromPage,  
                int nToPage,  
                int nExpType,  
                SPEXPORTPROC pCb,  
                LPVOID lpvParam);
```

Parameters

hDoc

Handle of document.

lpszFileName

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.

pCb

Points to the SPDEXPORTPROC function.

pvParam

User-defined data.

Return value

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

SPD_PrintDirect

Allows to send PDF data directly to the printer.

```
BOOL SPD_PrintDirect(HSPDFDOC hDoc,  
                    LPTSTR lpszPrinterName,  
                    int nFromPage,  
                    int nToPage,  
                    WORD wFlags);
```

Parameters

hDoc

Handle of document.

lpszPrinterName

Name of printer.

nFromPage

The first page in the range to be printed.

nToPage

The last page in the range to be printed.

wFlags

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.

SPV_Create

Creates a PDF-View window.

```
HWND SPV_Create(HWND hParentWnd,  
                LPCRECT lpRect,  
                UINT uID);
```

Parameters

hParentWnd

Handle of parent window.

lpRect

Specifies the size and position of the 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().

Messages

View window

It's true for every messages, that in case of process failure, the error code is accessible with the GetLastError() function.

WM_PD_ATTACH

Attaches a document handle to a window.

```
wParam = 0;           // not used
lParam = (HSPDFDOC) hDoc; // handle of the document
```

Parameters

hDoc
value of lParam. Document handle.

Return value

TRUE, in case of successful processing, otherwise FALSE.

WM_PD_DETACH

Detaches a document handle from a window.

```
wParam = 0; // not used
lParam = 0; // not used
```

Return value

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

WM_PV_SHOWPAGE

Displays a PDF page in the window.

```
wParam = nPage; // the page to display
lParam = fDPI; // the scale of resolution
```

Parameters

nPage
value of wParam. The number of the page to be displayed.

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

Return value

Number of the previously displayed page, in case of successful processing, otherwise 0.

WM_PV_ROTATEPAGE

Rotation of a PDF page.

```
wParam = nRotate; // measure of rotation
lParam = 0; // not used
```

Parameters

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

Return value

The value of the rotation of the previously displayed page, in case of successful processing, otherwise -1.

WM_PV_GETPAGESIZE

Getting the size of a page.

```
wParam = nPage; // the page to get the size of
lParam = fDPI;   // the scale of resolution
```

Parameters

nPage

value of wParam. The number of the page to get the size of.

fDPI

value of lParam. The scale of resolution of the page.

Return value

Address of the `SIZE` structure, in which the size of the actual displayed page is placed, in case of successful processing, otherwise 0.

WM_PV_GETSELTEXTLEN

Getting the number of characters of the selected text (if the selection exists)

```
wParam = 0; // not used
lParam = 0; // not used
```

Return value

Number of characters of the selection, or in case of failure, -1.

WM_PV_GETSELTEXT

Getting the content of a selected text

```
wParam = 0; // not used
lParam = lpzText; // points to the result in memory
```

Parameters

lpzText

value of lParam. Points to the memory buffer, where the content of the selected text is to be placed. The size of the buffer must be at least `SendMessage(hPdfWnd, WM_PV_GETSELTEXTLEN, 0, 0) + 1`.

Return value

TRUE, in case of successful processing, otherwise FALSE.

WM_PV_PRINT

Printing PDF page(s).

```
wParam = wPrintFlags; // printing settings
lParam = 0; // no used
```

Parameters

wPrintFlags

value of wParam. 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

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

WM_PV_FIND

Searching for text in the document.

wParam = uMsg; // indicator message
lParam = hWnd; // handle of receiving window

Parameters

uMsg
value of wParam. Value of the indicator message.
hWnd
value of lParam. Handle of the window receiving the indicator message.

Return value

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

WM_PV_DOC2WND

Converting document co-ordinates to window co-ordinates.

wParam = 0; // not used
lParam = lpptDoc; // points to a POINT structure.

Parameters

lpptDoc
value of lParam. Document co-ordinates.

Return value

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

WM_PV_WND2DOC

Converting window co-ordinates to document co-ordinates.

wParam = 0; // not used
lParam = lpptWnd; // points to a POINT structure.

Parameters

lpptWnd
value of lParam. Window co-ordinates.

Return value

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

Notifications

After the processing of a PDF command the window send the following notification messages to its parent window.

The parent window can process the notification in the WM_NOTIFY message.

Each notification message is called with a pointer of an NMSPV ACTION structure. The possible values of the 'code' member of the structure can be the following ones.

PVN_OPEN_FILE

Opening a PDF file.

```
nm.lParam = lpFileName;           // character string
```

Parameters

lpFileName

Value of nm.lParam. Points to the name of the file to be opened.

PVN_GOTO_PAGE

Go to a page.

```
nm.lParam = nPage;                // integer
```

Parameters

nPage

Value of nm.lParam. Page number.

Return value

In case of TRUE, displaying the actual page.

PVN_RUN_APP

Executing an application.

```
nm.lParam = lpzAppName;          // character string
```

Parameters

lpzAppName

Value of nm.lParam. The name of the application.

PVN_GO_BACK

Going to the previous state.

```
nm.lParam = 0;                   // not used
```

PVN_GO_FORWARD

Going to the next state.

```
nm.lParam = 0;                   // not used
```

PVN_QUIT

Quitting from the program.

```
nm.lParam = 0;                   // not used
```

Declarations

HSPDFDOC

Handle of a PDF document.

```
typedef HANDLE      HSPDFDOC;
```

SPDINFO

The **SPDINFO** structure defines the info-fields of a PDF file.

```
typedef struct tagSPDINFO
{
    char szTitle[SPDINFO_STR_SIZE];
    char szSubject[SPDINFO_STR_SIZE];
    char szKeywords[SPDINFO_STR_SIZE];
    char szProducer[SPDINFO_STR_SIZE];
    char szAuthor[SPDINFO_STR_SIZE];
    char szCreationDate[SPDINFO_STR_SIZE];
    char szCreator[SPDINFO_STR_SIZE];
    char szModDate[SPDINFO_STR_SIZE];
} SPDINFO, *PSPDINFO;
```

Members

szTitle

Title of the PDF document.

szSubject

Subject of the PDF document tárgya.

szKeywords

Keywords of the PDF document.

szProducer

Producer of the PDF document.

szAuthor

Author of the PDF document.

szCreationDate

Date of the PDF document's creation.

szCreator

Creator of the PDF document.

szModDate

Date of the PDF document's last modification.

NMSPVACTION

Structure for handling of the notification messages.

```
typedef struct tagNMSPVACTION
{
    NMHDR hdr;
```

```
    LPARAM IParam;  
} NMSPVACTION, *PNMSPVACTION;
```

Members

hdr

Information about the content of the notification message.

IParam

Parameter of the notification message.

SPDOUTLINEPROC

Function for processing bookmarks.

```
typedef BOOL (CALLBACK *SPDOUTLINEPROC)(LPSTR lpszTitle,  
    UINT nLevel,  
    DWORD dwID,  
    LPVOID lpvParam);
```

Parameters

lpszTitle

Title of the marker.

nLevel

Level of the marker in the tree.

dwID

Identifier of the marker.

lpvParam

Parameter.

SPDEXPORTPROC

Function for processing export.

```
typedef BOOL (CALLBACK *SPDEXPORTPROC)(int nPage,  
    LPVOID lpvParam);
```

Parameters

nPage

Processed page.

lpvParam

Parameter.

Constants

for the static arrays of the SPDINFO structure

SPDINFO_STR_SIZE

0xFF

The maximum length of the character string of the information structure (SPDINFO).

For printing

SPVPF_MODE_BMP	0x0001
Printing as a bitmap.	
SPVPF_MODE_PS	0x0002
Printing as a PostScript.	
SPVPF_PSF_LEVEL1	0x0008
Supporting the PS Level1 standard.	
SPVPF_PSF_LEVEL2	0x0010
Supporting the PS Level2 standard.	
SPVPF_PSF_LEVEL3	0x0020
Supporting the PS Level3 standard.	

Export types

SPD_EXPORT_TEXT	0x0001
Export as a text.	
SPD_EXPORT_BMP	0x0002
Export as a bitmap.	
SPD_EXPORT_XML	0x0003
Export as a xml.	

Error codes

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