Reference manual

for Sorax PDF SDK - ActiveX Edition

version 2.00

Contents

Features

Supported PDF version

PDF 1.7 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.

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.

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.

Return value

TRUE, if the processing was successful. 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.

GetPageLabel

Getting the label of a page.

BSTR GetPageLabel(OLE_HANDLE hDoc, LONG nPage);

Parameters

hDoc

Document handle.

nPage

The number of the page for which a label is needed.

Return value

A character string. The string represents the label of page.

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.

GetDocFontInfo

Processes the fonts of a PDF document.

VARIANT_BOOL GetDocFontToPageInfo(OLE_HANDLE hDoc,
LONG nFromPage,
LONG nToPage);

Parameters

hDoc
Handle of document.

nFromPage
The first page in the range.

nToPage
The last page in the range.

Return value

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

PrintDirect

Allows to send PDF data directly to the printer.

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

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.

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.

DoOpenAction

Execution of the PDF Document opening action (if there is one).

VARIANT_BOOL DoOpenAction(OLE_HANDLE hWnd);

Parameters

hWnd
Handle of the window.

Return value

TRUE, in case of successful processing, otherwise FALSE.

SetViewMode

Setting the mode of appearance for the view panel. In practice, it is nothing else, but the operation that realizes the change between the full screen and the normal view.

LONG SetViewMode(OLE_HANDLE hWnd, LONG nMode);

Parameters

hWnd
Handle of the window.

nMode
View mode.

Return value

The identifier of the view mode used prior to the setting or in the case of an error: 0.

SetViewLayout

Setting the layout used in the view panel.

LONG SetViewLayout(OLE_HANDLE hWnd, LONG nLayout);

Parameters

hWnd
Handle of the window.

nLayout
View layout.

Return value

The identifier of the layout used prior to the current setup or in the case of an error: 0.

SetActiveTool

Activating the current tool.

LONG SetActiveTool(OLE_HANDLE hWnd, LONG nTool);

Parameters

hWnd
Handle of the window.

nTool
Tool identifier.

Return value

The identifier of the tool that was used prior to the setup or in the case of an error: 0.

AutoScroll

Launching or stopping automatic scroll.

LONG AutoScroll(OLE_HANDLE hWnd, LONG nState, LONG nParam);

Parameters

hWnd

Handle of the window.

nState

Scroll command. Launching or stopping scroll, or query about its stage. For the possible values and their explanations, see the list of constants.

nParam

Time elapsed between two scrolls in the thousandth of a second. This value will determine the speed of the scrolling.

Return value

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

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. Posibble values: 0, 90, 180, 270.

SelectAll

Selecting all the textual content or cancelling any selection in the document.

VARIANT_BOOL SelectAll(OLE_HANDLE hWnd, VARIANT_BOOL bSelect);

Parameters

hWnd
Handle of the window.

bSelect
In the case of TRUE, all the textual content will become selected in the document.
In the case of FALSE, all the selections in the area become inactive.

Return value

TRUE, in case of successful processing, otherwise FALSE.

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.

GetPrevPage

Depending on the current layout it returns the number of the page previously displayed. If there is only one page, it returns the page previously displayed. In the case of a continuous layout the determination of the previous page in this view happens in relation to the (current) page that occupies the largest space.

LONG GetPrevPage(OLE_HANDLE hWnd, LONG nCurPage);

Parameters

hWnd

Handle of the window.

nCurPage

Current page. The number of the previous page will be determined in relation to this page.

Return value

The previous page number to be displayed, or in case of an error: 0.

GetNextPage

Depending on the current layout it returns the subsequent page number to be displayed, or if there is only one page, the subsequent page to be displayed. In the case of a continuous layout the determination of the subsequent page in this view happens in relation to the (current) page that occupies the largest space.

LONG GetNextPage(OLE_HANDLE hWnd, LONG nCurPage);

Parameters

hWnd

Handle of the window.

nCurPage

Current page.The number of the subsequent page will be specified in relation to this page.

Return value

The subsequent page number to be displayed, or in case of an error: 0.

DoOutlineAction

Hopping to the position specified in the Outline identifier.

LONG DoOutlineAction(OLE_HANDLE hWnd, ULONG ulID);

Parameters

hWnd

Handle of the window.

ulID

Outline identifier.

Return value

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

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.

ScrollPosChanged

The position of the document has been changed in the view panel.

void ScrollPosChanged(int nBar);

Parameters

nBar

The scroll bar constant that shows the direction of the displacement. Possible values SB_VSCROLL or SB_HSCROLL.

DocumentChanged

The displayed document of the window has been changed.

void DocumentChanged(BSTR lpszFileName);

Parameter

lpszFileName
File name with complete access path.

Searching

Searching in the document.

void Searching(LONG nPage, VARIANT_BOOL bRun);

Parameter

nPage
The page to search on.

bRun
Indicates the state of the searching.

InsertOutlineItem

Inserting a bookmark-item into the tree.

void InsertOutlineItem(LPCTSTR lpszTitle,
USHORT nLevel,
VARIANT_BOOL bExpanded,
ULONG ulID);

Parameter

lpszTitle
Title of the bookmark.

nLevel
Level to insert it.

bExpanded
Item is expanded or not.

luID
Identifier of the bookmark.

InsertFontInfoItem

Inserting a font information item.

void InserFontInfoItem(LPCTSTR lpszFontName,
ULONG ulFontFlags,
LPCTSTR lpszActFontName,
ULONG ulActFontFlags,
LONG nPage);

Parameter

lpszFontName
The name of the font.

ulFontFlags
Font flags. For the possible values and their explanations, see the list of constants.

lpszActFontName
Item is expanded or not.

ulActFontFlags
The applied Font flags. For the possible values and their explanations, see the list of constants.

nPage
The page number where the font can be found.

Constants

for the static arrays of the SPDINFO structure

SPDINFO_STR_SIZE **0xFF**

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

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.

Font info flags

SPD_FIF_TYPE_TYPE1 **0x0001**

Type1 font.

SPD_FIF_TYPE_TRUETYPE **0x0002**

TrueType font.

SPD_FIF_TYPE_EMBEDDED **0x0004**

Embedded font.

SPD_FIF_TYPE_SUBSET **0x0008**

Subset font.

SPD_FIF_TYPE_CID **0x0010**

CID font.

SPD_FIF_ENC_ANSI **0x0100**

ANSI encoding.

SPD_FIF_ENC_IDENTITYH **0x0200**

INDENTITYH encoding.

SPD_FIF_ENC_IDENTITYV **0x0400**

INDENTITYV encoding.

SPD_FIF_ENC_BUILTIN **0x0800**

BuiltIn encoding.

SPD_FIF_ENC_CUSTOM **0x1000**

Custom encoding.

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.

PVPF_PSF_SEPARATION **0x0080**

Separation.

PVPF_EMB_PSFONTS **0x0F00**

Fonts are embedded in the PS file.

PVPF_DUPLEX **0x4000**

Duplex printing.

PVPF_QUICKPRINT **0x8000**

Quick printing.

View layout

PVL_GETVIEWLAYOUT **0x00**

Query about the current layout.

PVL_SINGLEPAGE **0x01**

Single page layout.

PVL_TWOUP **0x02**

Tow-page layout.

PVL_CONTINUOUS **0x04**

Continuous layout.

PVL_SHOWHGAP **0x10**

Display of horizontal page gaps.

PVL_SHOWVGAP **0x20**

Display of vertical page gaps.

View mode

PVM_GETVIEWMODE **0x00**

Query for the mode of the current view.

PVM_NORMAL **0x01**

Activating normal view window.

PVM_FULLSCREEN **0x02**

Activating the full screen view window.

View scale

PVS_ZOOM **0**

Setting the value of the zoom.

PVS_ACTUALSIZE **-1**

Displays the pages in the original 100% size.

PVS_FITWIDTH **-2**

Aligns the extent of the zoom to the width of the window.

PVS_FITPAGE **-3**

Aligns the extent of the zoom to the full page display.

Auto scroll flags

PVASF_GETSTATE **0x00**

Query for the current state.

PVASF_SCROLL **0x01**

Launches or stops the scroll.

PVASF_DIR_UP **0x02**

Scrolls up the document.

PVASF_DIR_DOWN **0x04**

Scrolls down the document.

Cursor tools

PVCT_HAND **0x01**

Hand tool.

PVCT_SELECT **0x02**

Select tool.

PVCT_ZOOM_MARQUEE **0x03**

Marquee zoom tool.

PVCT_ZOOM_DYNAMIC **0x04**

Dynamic zoom tool.

PVCT_SNAPSHOT **0x05**

Making a snapshot.

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_EXITS **7005**

Nonexistent page.

ERROR_PERM_COPY **7006**

To copy to the clipboard is not permitted.

ERROR_PERM_PRINT **7007**

Printing is not permitted.

ERROR_PRINTER_INFO **7008**

Printer information cannot be queried.

ERROR_EXPORT_CCO **7021**

An object to be exported cannot be created.

ERROR_EXPORT_CIO **7022**

It is not possible to initialize the object to be exported.

ERROR_EXP_BMP_NEM **7031**

The memory is not enough for the execution of the export operation.

ERROR_EXP_BMP_FILECREATE **7032**

The file to be exported cannot be created.

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
.
.

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 able to find the extracted files in ElementsOf.<pdf_src> directory, whereabouts the pdf_src string is the title of PDF source file.