Reference manual

for

Sorax PDF SDK - DLL 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.

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_OpenBuf

Opens a PDF document from memory.

HSPDFDOC SPD_OpenBuf(LPSTR lpBuf,
UINT nLength,
LPSTR lpszUserPwd,
LPSTR lpszOwnerPwd);

Parameters

lpBuf
It points to the memory area that contains the PDF file.

nLength
The size of the memory area that contains the PDF file.

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_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 data 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 data 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_GetPageSize

Getting the size of a page.

BOOL SPD_GetPageSize(HSPDFDOC hDoc,
int nPage,
LPSIZE lpSize);

Parameters

hDoc
Handle of document.

nPage
The number of the page to get the size of.

lpSize
Address of the SIZE structure, in which the size of the actual displayed page is placed.

Return value

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

SPD_GetPageSizeEx

Getting the size of a page.

BOOL SPD_GetPageSizeEx(HSPDFDOC hDoc,
int nPage,
float fDPI,
LPSIZE lpSize);

Parameters

hDoc
Handle of document.

nPage
The number of the page to get the size of.

fDPI
The scale of resolution

lpSize
Address of the SIZE structure, in which the size of the actual displayed page is placed.

Return value

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

SPD_GetPageLabel

Getting the label of a page.

int SPD_GetPageLabel(HSPDFDOC hDoc,
int nPage,
LPSIZE lpszBuf,
size_t nSize);

Parameters

hDoc
Handle of document.

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

lpszBuf
Address of the label.

nSize
Size of buffer.

Return value

Length of the label string or 0, in case of an error. Error code is accessible with GetLastError().

SPD_GetVersion

Getting the version of the pdf document.

float SPD_GetVersion(HSPDFDOC hDoc);

Parameters

hDoc
Handle of document.

Return value

Version of the pdf document.

SPD_GetEncFlags

Getting the encrypt flags of the pdf document.

float SPD_GetEncFlags(HSPDFDOC hDoc);

Parameters

hDoc
Handle of document.

Return value

Encrypt flags of the pdf document.

SPD_GetPermFlags

Getting the permit flags of the pdf document.

float SPD_GetPermFlags(HSPDFDOC hDoc);

Parameters

hDoc

Handle of document.

Return value

Permit flags of the pdf document.

SPD_GetPageBitmap

On the basis of the data specified by the hDC device context handle it creates the device-independent bitmap for the given pdf page specified by the parameters.

HBITMAP SPD_GetPageBitmap(HSPDFDOC hDoc,
HDC hDC,
int nPage,
int nRot,
float fDPI);

Parameters

hDoc

Handle of document.

hDC

Device context handle.

nPage

The number of the page to get the bitmap for.

nRot

Value of rotation (0, 90, 180, 270).

fDPI

The sclae of resolution

Return value

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

SPD_GetFonts

Processes the fonts of a PDF document.

BOOL SPD_GetFonts(HSPDFDOC hDoc,
int nFromPage,
int nToPage,
SPDFONTINFOPROC pCb,
LPVOID pvParam);

Parameters

hDoc
Handle of document.

nFromPage
The first page in the range.

nToPage
The last page in the range.

pCb
Points to the SPDFONTINFOPROC function.

pvParam
User-defined data.

Return value

TRUE, if the processing was successful. FALSE, 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.

pvlParam
User-defined data.

Return value

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

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().

SPD_RegisterComp

Registration of the SPDF component.

void SPD_RegisterComp(LPSTR lpszUserName,
LPSTR lpszRegKey);

Parameters

lpszUserName
The user name that appears in the licence contract

lpszRegKey
The registration key that is given in the licence contract

SPD_ResetConfigString

Setting of the configurational character string.

BOOL SPD_ResetConfigString(LPSTR lpszNewVal,
UINT uKey);

Parameters

lpszNewVal
New configurational character string.

uKey
Key identifier.

Return value

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

SPD_ResetConfigInt

Setting the configuration value.

int SPD_ResetConfigInt(int nNewVal,
UINT uKey);

Parameters

nNewVal
New configuration value.

uKey
Key identifier.

Return value

The previous value to be set in case of successful processing, otherwise -1.

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 uesd
```

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_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.

```
wParam = nPage;   // Current page
lParam = 0;       // not used
```

Parameters

nPage

value of wParam. 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.

WM_PV_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.

```
wParam = nPage;   // Current page
lParam = 0;       // not used
```

Parameters

nPage

value of wParam. 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.

WM_PV_GETSCALE

Query for the current zoom value. For the list of the possible values, see: the list of constants.

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

Return value

The current zoom value, or in case of an error: 0

WM_PV_HASSELECTION

It shows if there is a selected area in the current view.

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

Return value

TRUE if there is a selected area in the current window, or FALSE if there is none. In case of an error: -1.

WM_PV_SELECTALL

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

wParam = bSelection; // Selection or cancel
lParam = 0; // not used

Parameters

bSelection

value of wParam. 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.

WM_PV_COPY

Copy of the selected area to the clipboard.

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

Return value

The number of the characters copied onto the clipboard, or in case of an error: -1.

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 wich the size of the actual displayed page is placed, in case of successful processing, otherwise 0.

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 = uMsgOrPar; // indicator message or parameter
lParam = hWndOrStr; // handle of receiving window or pointer to string

Parameters

uMsgOrPar
value of wParam. In the case of PVFP_GONEXT, hopping to the subsequent hit, in the case of (int)wParam > 0 the value of the indicator message.

hWndOrStr
value of lParam. In the case of wParam == PVFP_GETTEXT and wParam == PVFP_SETTEXT it points to a character string, in the case of wPartam > 0 it is the handle of the window that receives the message specified in wParam.

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

lpptDok
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.

WM_PV_DOOUTLINEACTION

Hopping to the position specified in the Outline identifier.

wParam = 0; // not used
lParam = lOutlineID; // Outline identifier

Parameters

lOutlineID

value oflParam. Outline identifier received in the argument of SPDOUTLINEPROC.

Return value

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

WM_PV_DOOPENACTION

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

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

Return value

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

WM_PV_ACTIVATETOOL

Activating the current tool.

wParam = wTool; // tool
lParam = 0; // no used

Parameters

wTool

value of wParam. Tool identifier. For the possible values and their explanation, see the list of constants.

Return value

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

WM_PV_SETVIEWLAYOUT

Setting the layout used in the view panel.

```
wParam = wLayout;       // View layout
lParam = 0;             // no used
```

Parameters

wViewLayout

value of wParam. Layout identifier. For the possible values and their explanation, see the list of constants.

Return value

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

WM_PV_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.

```
wParam = wViewMode;     // View mode
lParam = 0;             // not used
```

Parameters

wViewMode

value of wParam. For the possible values and their explanation, see the list of constants.)

Return value

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

WM_PV_BMIDTOPAGE

Query for a page related to a given Outline (bookmark) registry entry.

```
wParam = 0;             // no used
lParam = lOutlineID;    // Outline identifier
```

Parameters

lOutlineID

value oflParam. Outline identifier received in the argument of PDOUTLINEPROC.

Return value

The number of the page related to the registry entry, or in the case of an error: -1.

WM_PV_GETCURPAGE

Query for the current page.

wParam = 0; // no used
lParam = 0; // no used

Return value

In a one-page view it is unambiguously the page currently shown. If more than one page is visible at the same time in the view panel (in continuous and/or two-page presentation), then it will be the page that covers the largest area in the current page view. In the case of an error: 0.

WM_PV_GETDISPRECT

Query for a given page and the intersection of the rectangles specified by the view panel.

wParam = nPage; // page number,
lParam = lpRect; // points to the resulting rectangle intersection

Parameters

nPage

value of wParam. The number of the page for which we would like to receive the intersection with the view panel.

lpRect

value oflParam. The rectangle representing the intersection will be copied into this structure.

Return value

TRUE if the two rectangles have a real intersection, FALSE if they do not. In case of an error: -1.

WM_PV_SHOWDISPRECT

Displaying the part of a given page identified by a rectangle in the view panel.

wParam = nPage; // page number,
lParam = lpRect; // it points to the rectangle to be displayed

Parameters

nPage

value of wParam. The number of the page that has the part that we would like to see in the view panel.

lpRect

value oflParam. The rectangle that represents the part of the page to be displayed.

Return value

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

WM_PV_AUTOSCROLL

Launching or stopping automatic scroll.

```
wParam = wScroll;        // scroll command
lParam = lTime;          // time, in the thousandth of a second
```

Parameters

wScroll

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

lTime

value oflParam. 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.

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 NMSPVACTION 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_ PAGE_CHANGED

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_ SCROLLPOS_CHANGED

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

nm.lParam = nBar; // integer

Parameters

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

PVN_RUN_APP

Executing an application.

nm.lParam = lpszAppName; // character string

Parameters

lpszAppName
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

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 lParam;
} NMSPVACTION, *PNMSPVACTION;
```

Members

hdr
Information about the content of the notification message.

lParam
Parameters of the notification message.

SPDOUTLINEPROC

Function for processing bookmarks.

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

Parameters

lpszTitle
Title of the marker.

nLevel
Level of the marker in the tree.

dwID
Identifier of the marker.

bExpanded
Item is expanded or not.

lpvParam
Parameter.

SPDEXPORTPROC

Function for processing export.

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

Parameters

nPage
Processed page.

lpvParam
Parameter.

SPDFONTINFOPROC

Function for processing font information.

typedef BOOL (CALLBACK *SPDEXPORTPROC)(LPSTR lpszFontName,
    DWORD dwFontFlags,
    LPSTR lpszActFontName,
    DWORD dwActFontFlags,
    int nCurPage,
    LPVOID lpvParam);

Parameters

lpszFontName
The name of the font.

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

lpszActFontName
The name of the applied font.

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

nCurPage
The page number where the font can be found.

lpvParam
Parameter.

Constants

for the static arrays of the SPDINFO structure

SPDINFO_STR_SIZE **0x0400**

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

Encryption flags

SPD_ENF_OPEN_PWD **0x0001**

The document has an open password.

SPD_ENF_PERM_PWD **0x0002**

The document has an owner password.

SPD_ENF_ALGORITHM_RC4 **0x0010**

The document is coded using the RC4 encryption algorithm.

SPD_ENF_ALGORITHM_AES **0x0020**

The document is coded using the AES encryption algorithm.

SPD_ENF_KEYLENGTH_40 **0x0100**

The document is coded using a 40-bit encryption key.

SPD_ENF_KEYLENGTH_128 **0x0200**

The document is coded using a 128-bit encryption key.

Permission flags

SPD_PEF_PRINT **0x0004**

Printing is enabled in the document.

SPD_PEF_MODIFY **0x0008**

Modification of the document is enabled.

SPD_PEF_COPY **0x0010**

Copying to clipboard from the document is enabled.

SPD_PEF_COMMENT **0x0020**

Inserting annotations into the document is enabled.

SPD_PEF_FILL **0x0100**

Filling in forms within the document is enabled.

SPD_PEF_EXTRACT **0x0200**

Extraction of content from the document is enabled.

SPD_PEF_ASSEMBLY **0x0400**

Insertion of content into the document is enabled.

SPD_PEF_PRINTLOW **0x0800**

Only low resolution printing is permitted.

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.

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_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.

Color settings

Setting the colours for the view panel.

PVC_BKGND **0x01**

Setting the background colour.

PVC_PAPER **0x02**

Setting the colour of the paper.

PVC_PAPER_FRAME **0x03**

Setting the colour of the frame bordering the paper.

PVC_BORDER **0x04**

Setting the colour for the window frame.

PVC_FOCUS **0x05**

The colour of the window frame when it occupies the focus.

PVC_HOVER **0x06**

The colour of the frame when the mouse cursor is above the area of the window.

Window border

PVB_NONE **0x00**

No window frame will be drawn.

PVB_LEFT **0x01**

Only the left hand side of the window frame will be shown.

PVB_TOP **0x02**

Only the top of the window frame will be shown.

PVB_RIGHT **0x04**

Only the right hand side of the window frame will be shown.

PVB_BOTTOM **0x08**

Only the bottom of the window frame will be shown.

PVB_ALL **0x0F**

All of the window frame will be shown.

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.

Find parameters

PVFP_GONEXT **-1**

Finding the next hit.

PVFP_GETTEXT **-2**

Query for the current search text.

PVFP_SETTEXT **-3**

Setting current search text.

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.

Configuration identifiers

RCS_GEN_TEXTENCODING **0x01**

Setting the encoding of the text.

RCS_GEN_FONTPATH **0x02**

Setting the paths for the fonts.

RCI_EXP_BMP_DPI **0x01**

Setting the resolution of the bitmap to export.

RCI_EXP_XML_TEXT **0x02**

Showing or hiding the features of texts in the exported xml file.

RCI_EXP_XML_FONT **0x03**

Showing or hiding the features of fonts in the exported xml file.

RCI_EXP_XML_GRAPH **0x04**

Showing or hiding the features of graphic shapes in the exported xml file.

RCI_EXP_XML_EMBEDDEDIMAGE **0x05**

Permission, banning or marking of exporting embedded pictures in the xml file.

RCI_EXP_XML_EMBEDDEDFONT **0x06**

Permission, banning or marking of exporting embedded fonts in the xml file.

RCI_SMOOTH_TEXT **0x07**

Switches text smoothing on or off.

RCI_SMOOTH_LINEART **0x08**

Switches on or off the smoothing of graphic shapes.

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.