User Manual

Foxit PDF SDK (DLL) 2.0

Programming Guide
For Windows and Linux

Microsoft GOLD CERTIFIED Partner

Contents

Overview

Foxit PDF SDK, with minimized resource demand and redistribution size, is developed by Foxit Corporation for high-quality rendering of PDF document pages in a wide range of applications.

Foxit PDF SDK is specialized in PDF document rendering, with the capabilities of PDF parsing, editing, and creating.

Foxit PDF SDK supports almost all PDF page description features, except those very rarely used feature. The “Features” section lists all major features supported by the SDK.

Foxit PDF SDK is carefully designed and implemented so that for most documents and pages, it achieves a rendering speed comparable or faster than Adobe Acrobat Reader, while maintaining most of the same rendering quality.

Foxit PDF SDK runs on Windows 95/NT or later, and most of the functions also run on Linux (any recent releases). For detailed information on runtime configuration please refer to the redistribution section.

Evaluation Copy:

The downloaded version of Foxit PDF SDK from our website is for evaluation use only. It displays some small messages when rendering any PDF documents. Please note that the text module and the edit module will be unavailable when the evaluation period expires. If you are just evaluating the basic module (viewing), you don't need to call on the FPDF_UnlockDLL function. If you want to evaluate the text module and the edit module, you will have to call up this function. Please find the SN and UnlockCode in the Readme file. If you purchased a license of Foxit Reader SDK, Foxit will send a licensed copy of the SDK to you, and you need to call up the FPDF_UnlockDLL function to activate the interfaces that you purchased.

Features

Foxit PDF SDK (DLL) 2.0 does not only include the PDF viewing function (base module) in the last version, but also introduces two major functions – PDF text extracting (text module) and PDF creating/editing (edit module). The base module, coming with the head file names of " fpdfview.h and fpdfdoc.h", are used to render PDF documents, as well as obtain the navigation information from the PDF. If you want to use the text extraction function and the PDF editing function, please contact sales@foxitsoftware.com to purchase.

Here is a list of major PDF features supported by the basic module in the last version:

- PDF Specification: 1.3, 1.4, 1.5,1.6 and 1.7
- Automatic cross reference rebuild for documents damaged or generated by non-standard creator
- Compress Decoders: LZWDecode, FlateDecode, CCITTFaxDecode, DCTDecode
- Standard 40 bit and 128 bit encryption support
- All graphic objects: Path, Text, Image, Form, and Shading
- All color spaces: RGB, Gray, CMYK, ICCBased, Lab, Indexed, Separation, etc
- Tiling patterns and shadings
- Images: monochrome/colored image, JPEG images, masked and transparent images
- All font types: Type1, TrueType, Type0, Type3
- All CJK (Chinese/Japanese/Korean) character set support (additional redistribution file required)
- All text rendering modes: fill, stroke, or clipping
- Anti-aliasing of text rendering
- Transparent groups (including text, path, and image)
- Display annotations
- Unlimited page zoom factor
- Page rotation at 90, 180 and 270 degrees

Here is a list of major PDF features supported by FOXIT PDF SDK 2.0:

- Base Module:
  The base module, coming with the head file names of "pdfsdk.h and fpdfdoc.h," are used to render PDF documents and obtain the basic information of PDFs.
  a) Load PDFs from:
     - Standard File System
     - Memory
     - Custom source: non-standard file system, e.g. mobile phone and Internet
  b) Get PDF info, such as Document permission and Page count.
  c) Render PDF pages to:
     - Device (printer)

- Device Independent Bitmap

d) Get the info of bookmarks and links a PDF.

- Text Module:

  This module, coming with a head file name of “fpdftext.h,” has the functions listed below:

  a) Search text in one page or one document at a time
  b) Extract text from a designated rectangle area or from an indices-range.
  c) Convert a PDF document or a single page to a text file.
  d) Obtain the information for each character in a PDF, including the Unicode code, font, font size, and the location information

- Edit Module:

  This module, coming with a head file name of “ fpdfedit.h,” has the functions listed below:

  a) Create a PDF document.
  b) Create new pages or deletes existing pages from a PDF.
  c) Extract all kinds of objects included in a PDF, such as text object, path object, image object and form object.
  d) Modifies the content of a PDF document. Users can add, delete or edit any objects that are included in the PDF.

For the detailed information of the interfaces, please refer to Function Reference.

Tutorials

In this section we provide easy to follow tutorials on how to use FOXIT PDF SDK to render PDF pages as you like. Please note we use C language in all tutorials, so if you are using other programming languages, please refer to the corresponding information on how you can call the DLL. For C/C++ language, you can include the FPDFVIEW.H head file in your source code, and call the functions listed in the head file. You also need to link with the FPDFSDK.LIB file (on Windows) or libfpdfsdk.so (on Linux).

FOXIT PDF SDK is a standalone dynamic library (DLL on Windows or shared library on Linux), it doesn't require any runtime language support (except basic Windows or Linux components), and therefore, you can call the SDK from any programming language which supports Windows system function calls or Linux shared library calls.

FOXIT PDF SDK uses only basic data types including integer, double, zero-terminated string, and opaque pointers. You can find any of these types in any programming language. NOTE: for opaque pointer types defined in fpdfview.H head file, including PDF_DOCUMENT and PDF_PAGE, can be treated simply as 32 bit integers.

Most programming languages support two types of DLL calling:

1. **Static loading**: you can declare the functions in your own language, at the compile time, so when the program starts, it will automatically look for FOXIT PDF SDK and load all functions;
2. **Dynamic loading**: you can call Windows system functions to load the DLL at any time after your program starts, and then you can load the necessary functions from the DLL, and call them.

TUTORIAL 1: Initialize and Destroy the FPDFSDK library

You have to call the FPDF_InitLibrary function before you can call any PDF processing functions and to call FPDF_DestroyLibrary to release all memory blocks allocated by the library.

```
#include "fpdfview.h"
…
FPDF_InitLibrary(this->m_hInstance); // Initialize the library.
…                                    // Call some function from the DLL.
FPDF_DestroyLibrary();               // After this function called, you should
                                     // not call any PDF processing functions.
```

TUTORIAL 2: Load PDF Document and Page

In this tutorial, we will load a PDF document named “testdoc.pdf”, and then load the first page of the document. In order to render any page, the page must be loaded first.

```
#include “fpdfview.h”
…
FPDF_DOCUMENT pdf_doc;
FPDF_PAGE pdf_page;

// Before we can do anything, we need to unlock the DLL
// NOTE: If you are evaluating FOXIT PDF SDK, you don’t need unlock the DLL,
// then evaluation marks will be shown with all rendered pages.
FPDF_UnlockDLL(“license_id”, “unlock_code”);
// first, load the document (no password specified)
pdf_doc = FPDF_LoadDocument(“testdoc.pdf”, NULL);
if (pdf_doc == NULL) … // error handling
// now load the first page (page index is 0)
pdf_page = FPDF_LoadPage(pdf_doc, 0);
if (pdf_page == NULL)
…      // error handling
```

TUTORIAL 3: Basic Rendering on Windows

In this tutorial, we will render a loaded page onto a screen window, within a specified screen rectangle area. Note we don’t show you the whole program here, only the handling of WM_PAINT message is shown. You need to put the code into the right place according to the development tool you choose to use.

Let us assume we have a handle to the window, which is stored in hWnd variable.

```
…
HWND hWnd;
FPDF_DOCUMENT pdf_doc; // you must load document before page can be loaded
FPDF_PAGE pdf_page; // you must load page before it can be rendered
…
// Get handle to Device Context (DC) for the window
HDC hDC = GetDC(hWnd);
// Call FPDF_RenderPage function to render the whole page
FPDF_RenderPage(hDC, pdf_page, 10, 10, 400, 500, 0, 0);
```

The codes above will render the whole page onto the window, with the left-top corner of the page at point {10, 10} (coordinate relative to the left-top corner of the window), and the page will be fit into a rectangle with the

width of 400 pixels and height of 500 pixels. The page will not be rotated.

TUTORIAL 4: Basic Rendering on Linux

Foxit PDF SDK doesn't directly support X-Windows output, however, you can use the Foxit Device Independent Bitmap (FXDIB) features provided in the SDK to render any PDF page into a bitmap device. You can then transfer the bitmap onto X-Windows surface, or image file, or other type of devices.

Before you can use a bitmap device, you need to create one and you can use the FPDFBitmap_Create function to do that. Foxit PDF SDK supports two types of bitmaps: RGB bitmap and ARGB bitmap, you can choose which format you want to use when you create the bitmap.

When the bitmap is created, you need to fill the buffer with the background that you like. Normally it can be a white background or a non-colored background (if alpha channel is used).

Now you can use it in the FPDF_RenderPageBitmap function. This function is pretty much the same as the FPDF_RenderPage function, but instead of sending output to a Windows device, it outputs to the bitmap device.

Here is a sample for basic rendering on Linux:

```
FPDF_BITMAP bitmap;
FPDF_DOCUMENT pdf_doc;  // you must load document before page can be loaded
FPDF_PAGE pdf_page; // you must load page before it can be rendered
…
// Create bitmap first. We don't use alpha channel here.
bitmap = FPDFBitmap_Create(400, 500, FALSE);

// Fill the bitmap with white background
FPDFBitmap_FillRect(bitmap, 0, 0, 400, 500, 255, 255, 255, 255);

// Call FPDF_RenderPageBitmap function to render the whole page
FPDF_RenderPageBitmap(bitmap, pdf_page, 10, 10, 400, 500, 0, 0);
```

The following tutorials introduce more features of rendering in Foxit PDF SDK. All the samples use the FPDF_RenderPage function however the same techniques apply to FPDF_RenderPageBitmap, in the same way.

TUTORIAL 5: Zoom the Page

In tutorial 2 we rendered the page into a fixed size rectangle. This is simple; however, it's not that useful. In most cases, you want to show the PDF page in its original size, or zoom in/zoom out for larger/smaller display. All these operations can be achieved by adjustment of the size parameters (the fifth and sixth parameters) of FPDF_RenderPage function.

If you want to show the page in its original size, you first need to know the original size of the page, this is easy, and you can call one of the following functions to get width and height of a page:

```
double page_width, page_height;
…
page_width = FPDF_GetPageWidth(pdf_page);
page_height = FPDF_GetPageHeight(pdf_page);
```

Please note the page width and height returned from the above two functions and are measured in typographic POINTS. In typography, one point equals to 1/72 inch, which is about 0.35 milli-miter. You can not directly pass the width or height in points into FPDF_RenderPage function, because FPDF_RenderPage accepts only pixels for screen size. Fortunately you can convert the points to pixels if you know the size of one pixel on your computer screen:

```
int logpixelsx, logpixelsy, size_x, size_y;
HDC hDC;
…
// get number of pixels per inch (horizontally and vertically)
logpixelsx = GetDeviceCaps(hDC, LOGPIXELSX);
logpixelsy = GetDeviceCaps(hDC, LOGPIXELSY);
// convert points into pixels
Size_x = page_width / 72 * logpixelsx;
size_y = page_height / 72 * logpixelsy;
```

After you get the original page size in pixels, you can use it in the FPDF_RenderPage to display the page in its original size:

```
…
FPDF_RenderPage(hDC, pdf_page, 10, 10, size_x, size_y, 0, 0);
```

Now, you may want to zoom in or out of the page on the screen. Note you can zoom both vertically and in a horizontal direction, with separate zoom factors. Again, all you need to do is to adjust the size_x and size_y parameter when you call the FPDF_RenderPage function.

The following call shows the page with 200% zoom factor:

```
…
FPDF_RenderPage(hDC, pdf_page, 10, 10, size_x * 2, size_y * 2, 0, 0);
```

While the following call shrink the page by 50%:

```
…
FPDF_RenderPage(hDC, pdf_page, 10, 10, size_x / 2, size_y / 2, 0, 0);
```

TUTORIAL 6: Scroll the Page

Sometimes your PDF viewing codes need to provide scroll capability, especially when the page is zooming into a large size exceeding the displayable area in the screen window. To scroll the page to a different position, horizontally or vertically, you can adjust the start_x and start_y parameters when you call FPDF_RenderPage function.

Because start_x and start_y parameters always specify the position of the top-left corner for displaying the page, so when you scroll down, the top-left corner of the page will go up, therefore the start_y parameter should be decreased. Sometimes the parameter can become negative, this is OK, this is a way to tell the FPDF_RenderPage function that the top-left corner of the page is not visible any more. Likewise, when you scroll up, the start_y will increase. When you scroll left/right, the start_x parameters will increase/decrease accordingly.

Here is an example how you can scroll the page vertically to half of its height:

```
int start_y;
…
// Calculate the new start_y, decreased by the scroll pixel size
start_y = 10 - page_height / 2;
FPDF_RenderPage(hDC, pdf_page, 10, start_y, size_x, size_y, 0, 0);
```

TUTORIAL 7: Rotate the Page

PDF pages can be displayed upright like it is, or, it can be rotated in both direction (clockwise or counter-clockwise), or it can be displayed upside-down. To achieve these different rotations, you just need to adjust the rotate parameter when you call FPDF_RenderPage function.

The rotate parameter can be 0, 1, 2, or 3. Use 0 for no rotated,1 for rotated (90 degrees) clockwise, 2 for

upside-down, 3 for rotated (90 degrees) counter-clockwise.

The following example shows how you can rotate the page clockwise:

```
…
FPDF_RenderPage(hDC, pdf_page, 10, 10, size_x, size_y, 1, 0);
```

TUTORIAL 8: Render Part of the Page

Very often you don't want to re-render the whole page because it might be time consuming, you just want to re-render the area that needs to be re-painted. FPDF_RenderPage supports partial rendering through the standard Windows clip box mechanism,that is, if you have clip box correctly set, FPDF_RenderPage automatically render only page objects (texts, graphics, or images) inside the clip box.

You can set the clip box implicitly using WIN32 function InvalidateRect, or, call one of the clipping WIN32 functions (like SelectClipPath) to set the clip box explicitly.

Here is an example showing how you can call InvalidateRect to set a clip box, so only the partial page will be re-rendered. We assume you have handled WM_PAINT message correctly by calling FPDF_RenderPage function.

```
RECT rect;
int left, right, top, bottom;
…
rect.left = left;
rect.right = right;
rect.top = top;
rect.bottom = bottom;
InvalidateRect(hWnd, &rect, TRUE);
```

Note: for correctly re-render the page content; you MUST clear the drawing area first, as we indicated in the last parameter of InvalidateRect function call. And you need to handle WM_ERASEBKGND function to fill the background with pure white.

TUTORIAL 9: Print Pages

To output a PDF pages to the printer it is almost the same as displaying the pages on the screen. Windows system treats both screen and printer as display devices. So you can get a DC (Device Context) for a printer, and then you can use the same FPDF_RenderPage function to output to the printer.

The only difference would be for printing you need to deal with documents and pages. We don't discuss how to get a printer DC here, please refer to WIN32 documents or documents about your development tool.

Here is an example on how you can print a loaded page:

```
HDC hDC;
DOCINFO doc_info;
…
// Get a printer DC
hDC = CreateDC("WINSPOOL", "Printer Name", NULL, NULL);
// Start a printer job
StartDoc(hDC, &doc_info);
// Start a new printing page
StartPage(hDC);
…  // we need to calculate the page size here
// Now we can render the page to the printer
FPDF_RenderPage(hDC, pdf_page, 0, 0, size_x, size_y, 0, 0);
…
```

TUTORIAL 10: Convert Page into Bitmap

Another application of FPDF_RenderPage function is converting a page into a bitmap. Windows allows you can create a memory DC which holds a bitmap inside, when you draw onto that DC nothing has been outputted, except the bitmap has been changed. Therefore, after FPDF_RenderPage finished its rendering into a bitmap DC, you can take the bitmap anywhere, and save it into an image file, with any format you may like.

Here is an example:

```
HDC hDC, hMemDC;
HBITMAP hBitmap;
…
// Create a memory DC, which is compatible with the screen DC
hMemDC = CreateCompatibleDC(hDC);
// Create a bitmap for output
hBitmap = CreateCompatibleBitmap(hDC, size_x, size_y);
// Select the bitmap into the memory DC
hBitmap = (HBITMAP)SelectObject(hMemDC, hBitmap);
// Now render the page onto the memory DC, which in turn changes the bitmap
FPDF_RenderPage(hMemDC, pdf_page, 0, 0, size_x, size_y, 0, 0);
```

```
// Release the bitmap from the memory DC
hBitmap = (HBITMAP)SelectObject(hMemDC, hBitmap);
// Now the bitmap is filled with page contents.
// You can save it using WIN32 or other tools
…
```

TUTORIAL 11: Search on a page

FPDFTEXT module is quite an important function for PDF searching. To use this module, please call the functions of FPDFText_FindStart, FPDFText_FindNext and FPDFText_FindPrev.

Here is an example:

```
// Get the text page from a pdf page.
FPDF_TEXTPAGE pTextPage = FPDFText_LoadPage(pdf_Page);
// Set the find flag
int nFlag = FPDF_MATCHCASE | FPDF_MATCHWHOLEWORD;
// Get the search context handle
FPDF_SCHHANDLE pSCHHandle = FPDFText_FindStart(pTextPage, findtext, nFlag, 0);
// Search the text string.
BOOL bResult = FPDFText_FindNext(m_pSCHHandle);
…

// Release the search context
FPDFText_FindClose(pSCHHandle);
// Release all resources allocated for a text page information structure
FPDFText_ClosePage(pTextPage);
```

TUTORIAL 12: Create a new pdf document

FPDFEDIT module is used to add, delete and edit a PDF document. To create a PDF document, please call the functions of FPDF_CreateNewDocument , FPDFPage_New. FPDF_CreateNewDocument is used to create a blank PDF document, whose page count is 0. To edit the page, please call FPDFPage_New function to add pages first.

Here is an example:

```
// Create an empty pdf document
FPDF_DOCUMENT pPDFDoc = FPDF_CreateNewDocument();
// Insert a pdf page
FPDF_PAGE pdf_page = FPDFPage_New(pPDFDoc,0,500,600);

// New an image object
FPDF_PAGEOBJECT page_obj = FPDFPageObj_NewImgeObj(pPDFDoc);
```

```
// Load the image from file
BOOL ret = FPDFImageObj_LoadFromFile(pdf_page,1,page_obj,file_path);
// Insert a pdf page
FPDFPage_InsertObject(pdf_page,page_obj);
…

// Save the new document
FPDF_SaveAsFile(pPDFDoc,filepath,NULL,NULL,0,NULL,0);
```

Function Reference

Details of functions in Foxit PDF SDK are removed from this section. For latest information on each function, please refer to the header file.

1. FPDFVIEW Operation (Base Module)

FPDF_InitLibrary

Initialize the FPDFSDK library.

Prototype:

void FPDF_InitLibrary(void* hInstance);

Parameters:

hInstance - For WIN32 system only: the instance of the executable or DLL module

Return value:

None.

Comments:

You have to call this function before you can call any PDF processing functions.

FPDF_GetModuleMgr

Get the module of this DLL.

Prototype:

FPDF_MODULEMGR FPDF_GetModuleMgr();

Parameters:

None.

Return value:

FPDF_MODULEMGR - The handle of this module.

FPDF_DestroyLibary

Release all resources allocated by the FPDFSDK library.

Prototype:

Void FPDF_DestroyLibrary();

Parameters:

None.

Return value:

None.

Comments:

You can call this function to release all memory blocks allocated by the library.
After this function called, you should not call any PDF processing functions.

FPDF_UnlockDLL

Unlock the DLL using license key info received from Foxit

Prototype:

```
void    FPDF_UnlockDLL(FPDF_BYTESTRING license_id,
                       FPDF_BYTESTRING unlock_code);
```

Parameters:

license_id - A string received from Foxit identifying the SDK license.
unlock_code - A string received from Foxit for unlocking the DLL.

Return value:

None.

Comments:

For SDK evaluators, this function call is not required, then all rendered pages will come with an evaluation mark. For purchased SDK customers, this should be the first function to call before any other functions to be called.

FPDF_LoadDocument

Open and load a PDF document.

Prototype:

```
FPDF_DOCUMENT  FPDF_LoadDocument(FPDF_STRING file_path,
                                 FPDF_BYTESTRING password);
```

Parameters:

file_path - Path to the PDF file (including extension)
password - A string used as the password for PDF file.
If no password needed, empty or NULL can be used.

Return value:

A handle to the loaded document. If failed, NULL is returned.

Comments:

Loaded document can be closed by FPDF_CloseDocument.
If this function fails, you can use FPDF_GetLastError() to retrieve the reason why it fails.

FPDF_LoadMemDocument

Open and load a PDF document from memory.

Prototype:

```
FPDF_DOCUMENT  FPDF_LoadMemDocument(const void* data_buf,
                                    int size, FPDF_BYTESTRING password);
```

Parameters:

data_buf - Pointer to a buffer containing the PDF document.

size - Number of bytes in the PDF document.
password - A string used as the password for PDF file.
If no password needed, empty or NULL can be used.

Return value:

A handle to the loaded document. If failed, NULL is returned.

Comments:

The memory buffer must remain valid when the document is open. Loaded document can be closed by FPDF_CloseDocument. If this function fails, you can use FPDF_GetLastError() to retrieve the reason why it fails.

```
// Structure for custom file access
typedef struct {
    // File length, in bytes
    unsigned long      m_FileLen;

    // A function pointer for getting a block of data from specific position.
    // Position is specified by byte offset from beginning of the file.
    // The position and size will never go out range of file length.
    // It may be possible for FPDFSDK to call this function multiple times for same position.
    // Return value: should be non-zero if successful, zero for error.
    int                (*m_GetBlock)(void* param, unsigned long position, unsigned char* pBuf, unsigned long size);

    // A custom pointer for all implementation specific data.
    // This pointer will be used as the first parameter to m_GetBlock callback.
    void*              m_Param;
} FPDF_FILEACCESS
```

FPDF_LoadCustomDocument

Load PDF document from a custom access descriptor.

Prototype:

```
FPDF_DOCUMENT    FPDF_LoadCustomDocument(FPDF_FILEACCESS* pFileAccess,
                                         FPDF_BYTESTRING password);
```

Parameters:

pFileAccess - A structure for access the file.
password - Optional password for decrypting the PDF file.

Return value:

A handle to the loaded document. If failed, NULL is returned.

Comments:

The file resources should stay valid until the PDF document is closed.

Loaded document can be closed by FPDF_CloseDocument.

FPDF_GetLastError

Get last error code when an SDK function failed.

Prototype:

unsigned long FPDF_GetLastError();

Parameters:

None.

Return value:

A 32-bit integer indicating error codes (defined below).

0 no error
1 unknown error
2 file not found or could not be opened
3 file not in PDF format or corrupted
4 password required or incorrect password
5 unsupported security scheme
6 page not found or content error

Comments:

If the previous SDK call succeeded, the return value of this function is not defined.

FPDF_GetDocPermission

Get file permission flags of the document.

Prototype:

unsigned long FPDF_GetDocPermissions(FPDF_DOCUMENT document);

Parameters:

document - Handle to document. Returned by FPDF_LoadDocument function.

Return value:

A 32-bit integer indicating permission flags. Please refer to PDF Reference for detailed description. If the document is not protected, 0xffffffff will be returned.

FPDF_GetPageCount

Get total number of pages in a document.

Prototype:

int FPDF_GetPageCount(FPDF_DOCUMENT document);

Parameters:

document - Handle to document. Returned by FPDF_LoadDocument function.

Return value:

Total number of pages in the document.

FPDF_LoadPage

Load a page inside a document.

Prototype:

FPDF_PAGE FPDF_LoadPage(FPDF_DOCUMENT document, int page_index);

Parameters:

document - Handle to document. Returned by FPDF_LoadDocument function.

page_index - Index number of the page. 0 for the first page.

Return value:

A handle to the loaded page. If failed, NULL is returned.

Comments:

Loaded page can be rendered to devices using FPDF_RenderPage function.

Loaded page can be closed by FPDF_ClosePage.

FPDF_GetPageWidth

Get page width

Prototype:

double FPDF_GetPageWidth(FPDF_PAGE page);

Parameters:

page - Handle to the page. Returned by FPDF_LoadPage function.

Return value:

Page width (excluding non-displayable area) measured in points.

One point is 1/72 inch (around 0.3528 mm)

FPDF_GetPageHeight

Get page height

Prototype:

double FPDF_GetPageHeight(FPDF_PAGE page);

Parameters:

page - Handle to the page. Returned by FPDF_LoadPage function.

Return value:

Page height (excluding non-displayable area) measured in points.

One point is 1/72 inch (around 0.3528 mm)

FPDF_ GetPageSizeByIndex

Get the size of a page by index.

Prototype:

int FPDF_GetPageSizeByIndex(FPDF_DOCUMENT document, int page_index,
double* width, double* height);

Parameters:

document - Handle to document. Returned by FPDF_LoadDocument function.
page_index - Page index, zero for the first page.
width - Pointer to a double value receiving the page width (in points).
height - Pointer to a double value receiving the page height (in points).

Return value:

Non-zero for success. 0 for error (document or page not found).

FPDF_EnumPageSize

Enumerating all pages within the document.

Prototype:

```
void FPDF_EnumPageSize(FPDF_DOCUMENT document,
    FPDF_ENUMPAGESIZEPROC callback);
```

Parameters:

document - Handle to document. Returned by FPDF_LoadDocument function.
callback - A pointer to a callback function,

```
void (*FPDF_ENUMPAGESIZEPROC)(int page_index, double width, double height)
```

Parameters:

page_index - Page index, zero for the first page.
width - The page width.
height - The page height.

Return value:

None.

Return value:

None.

FPDF_RenderPage

Render contents in a page to a device (screen, bitmap, or printer).
This function is only supported on Windows system.

Prototype:

```
void FPDF_RenderPage(HDC dc, FPDF_PAGE page, int start_x, int start_y, int size_x,
    int size_y,  int rotate, int flags);
```

Parameters:

dc - Handle to device context.
page - Handle to the page. Returned by FPDF_LoadPage function.
start_x - Left pixel position of the display area in the device coordinate.
start_y - Top pixel position of the display area in the device coordinate.
size_x - Horizontal size (in pixels) for displaying the page.
size_y - Vertical size (in pixels) for displaying the page.
rotate - Page orientation: 0 (normal), 1 (rotated 90 degrees clockwise),
2 (rotated 180 degrees), 3 (rotated 90 degrees counter-clockwise).

flags - 0 for normal display, or combination of flags defined below

0x01	Set if annotations are to be rendered
0x02	Set if using text rendering optimized for LCD display
0x04	Set if you don't want to use GDI+ Applicable to desktop Windows systems only.
0x08	Grayscale output
0x80	Set if you want to get some debug info.
0x100	Set if you don't want to catch exception

Return value:

None.

FPDF_RenderPageBitmap

Render contents in a page to a device-independent bitmap

Prototype:

void FPDF_RenderPageBitmap(FPDF_BITMAP bitmap, FPDF_PAGE page, int start_x, int start_y, int size_x, int size_y, int rotate, int flags);

Parameters:

bitmap - Handle to the device independent bitmap (as the output buffer). Bitmap handle can be created by FPDFBitmap_Create function.

page - Handle to the page. Returned by FPDF_LoadPage function.

start_x - Left pixel position of the display area in the device coordinate

start_y - Top pixel position of the display area in the device coordinate

size_x - Horizontal size (in pixels) for displaying the page

size_y - Vertical size (in pixels) for displaying the page

rotate - Page orientation: 0 (normal), 1 (rotated 90 degrees clockwise), 2 (rotated 180 degrees), 3 (rotated 90 degrees counter-clockwise).

flags - 0 for normal display, or combination of flags defined above

Return value:

None.

FPDF_QuickDrawPage

Draw a thumbnail of a page into a bitmap

Prototype:

void FPDF_QuickDrawPage(FPDF_BITMAP bitmap, FPDF_PAGE page, int start_x, int start_y, int size_x, int size_y, int rotate, int flags);

Parameters:

bitmap - Handle to the device-independent bitmap (as the output buffer). Bitmap handle can be created by FPDFBitmap_Create function.

page - Handle to the page. Returned by FPDF_LoadPage function.

start_x - Left pixel position of the display area in the device coordinate

start_y - Top pixel position of the display area in the device coordinate
size_x - Horizontal size (in pixels) for displaying the page
size_y - Vertical size (in pixels) for displaying the page
rotate - Page orientation: 0 (normal), 1 (rotated 90 degrees clockwise), 2 (rotated 180 degrees), 3 (rotated 90 degrees counter-clockwise).
flags - Currently must be zero.

Return value:

None.

Comments:

This functions draws a very low-resolution thumbnail of a page, sometimes with inaccurate shape or position. The result thumbnail is meant for a very rough preview of the page contents, just giving user some idea about how the page looks like. The thumbnail is often useful in multi-threaded or progressive environment, the application can first display the low-resolution thumbnail, start to respond to user input, and a higher resolution thumbnail can be generated by FPDF_RenderPageBitmap function.

FPDF_ClosePage

Close a loaded PDF page.

Prototype:

```
void FPDF_ClosePage(FPDF_PAGE page);
```

Parameters:

page - Handle to the loaded page

Return value:

None.

FPDF_CloseDocument

Close a loaded PDF document.

Prototype:

```
void FPDF_CloseDocument(FPDF_DOCUMENT document);
```

Parameters:

document - Handle to the loaded document

Return value:

None.

FPDF_DeviceToPage

Convert the screen coordinate of a point to page coordinate.

Prototype:

```
void FPDF_DeviceToPage(FPDF_PAGE page, int start_x, int start_y, int size_x, int size_y,
    int rotate, int device_x, int device_y, double* page_x, double* page_y);
```

Parameters:

page	-	Handle to the page. Returned by FPDF_LoadPage function.
start_x	-	Left pixel position of the display area in the device coordinate
start_y	-	Top pixel position of the display area in the device coordinate
size_x	-	Horizontal size (in pixels) for displaying the page
size_y	-	Vertical size (in pixels) for displaying the page
rotate	-	Page orientation: 0 (normal), 1 (rotated 90 degrees clockwise), 2 (rotated 180 degrees), 3 (rotated 90 degrees counter-clockwise).
device_x	-	X value in device coordinate, for the point to be converted
device_y	-	Y value in device coordinate, for the point to be converted
page_x	-	A pointer to a double value receiving the converted X value in page coordinate
page_y	-	A pointer to a double value receiving the converted Y value in page coordinate

Return value:

None.

Comments:

The page coordinate system has its origin at left-bottom corner of the page, with X axis goes along the bottom side to the right, and Y axis goes along the left side upward. NOTE: this coordinate system can be altered when you zoom, scroll, or rotate a page, however, a point on the page should always have the same coordinate values in the page coordinate system.

The device coordinate system is device dependant. For screen device, its origin is at left-top corner of the window. However this origin can be altered by Windows coordinate transformation utilities. You must make sure the start_x, start_y, size_x, size_y and rotate parameters have exactlysame values as you used in FPDF_RenderPage() function call.

FPDF_PageToDevice

Convert the page coordinate of a point to screen coordinate.

Prototype:

```
void FPDF_PageToDevice(FPDF_PAGE page, int start_x, int start_y, int size_x, int size_y,
    int rotate, double page_x, double page_y, int* device_x, int* device_y);
```

Parameters:

page	-	Handle to the page. Returned by FPDF_LoadPage function.
start_x	-	Left pixel position of the display area in the device coordinate
start_y	-	Top pixel position of the display area in the device coordinate
size_x	-	Horizontal size (in pixels) for displaying the page
size_y	-	Vertical size (in pixels) for displaying the page
rotate	-	Page orientation: 0 (normal), 1 (rotated 90 degrees clockwise), 2 (rotated 180 degrees), 3 (rotated 90 degrees counter-clockwise).
page_x	-	X value in page coordinate, for the point to be converted

page_y - Y value in page coordinate, for the point to be converted
device_x- A pointer to an integer value receiving the result X value in device coordinate.
device_y- A pointer to an integer value receiving the result Y value in device coordinate.

Return value:

None

Comments:

See comments of FPDF_DeviceToPage() function.

FPDFBitmap_Create

Create a Foxit Device-Independent Bitmap (FXDIB)

Prototype:

FPDF_BITMAP FPDFBitmap_Create(int width, int height, int alpha);

Parameters:

width - Number of pixels in a horizontal line of the bitmap. Must be greater than 0.
height - Number of pixels in a vertical line of the bitmap. Must be greater than 0.
alpha - A flag indicating whether alpha channel is used. Non-zero for using alpha, zero for not using.

Return value:

The created bitmap handle, or NULL if parameter error or out of memory.

Comments:

An FXDIB always use 4 byte per pixel. The first byte of a pixel is always double word aligned. Each pixel contains red (R), green (G), blue (B) and optionally alpha (A) values. The byte order is BGRx (the last byte unused if no alpha channel) or BGRA. The pixels in a horizontal line (also called scan line) are stored side by side, with left most pixel stored first (with lower memory address). Each scan line uses width*4 bytes. Scan lines are stored one after another, with top most scan line stored first. There is no gap between adjacent scan lines.

This function allocates enough memory for holding all pixels in the bitmap, but it doesn't initialize the buffer. Applications can use FPDFBitmap_FillRect to fill the bitmap using any color.

FPDFBitmap_CreateEx

Create a Foxit Device-Independent Bitmap (FXDIB)

Prototype:

FPDF_BITMAP FPDFBitmap_CreateEx(int width, int height, int format, void* first_scan,
int stride);

Parameters:

width - Number of pixels in a horizontal line of the bitmap. Must be greater than 0.
height - Number of pixels in a vertical line of the bitmap. Must be greater than 0.
format - A number indicating for bitmap format, as defined below
1 Gray scale bitmap, one byte per pixel

2 3 bytes per pixel, byte order: blue, green, red
3 4 bytes per pixel, byte order: blue, green, red, unused
4 4 bytes per pixel, byte order: blue, green, red, alpha

first_scan - A pointer to the first byte of first scan line, for external buffer only. If this parameter is NULL, then the SDK will create its own buffer.

stride - Number of bytes for each scan line, for external buffer only..

Return value:

The created bitmap handle, or NULL if parameter error or out of memory.

Comments:

Similar to FPDFBitmap_Create function, with more formats and external buffer supported. Bitmap created by this function can be used in any place that a FPDF_BITMAP handle is required. If external scanline buffer is used, then the application should destroy the buffer by itself. FPDFBitmap_Destroy function will not destroy the buffer.

FPDFBitmap_FillRect

Fill a rectangle area in an FXDIB

Prototype:

```
void FPDFBitmap_FillRect(FPDF_BITMAP bitmap, int left, int top, int width, int height,
    int red, int green, int blue, int alpha);
```

Parameters:

bitmap - The handle to the bitmap. Returned by FPDFBitmap_Create function.

left - The left side position. Starting from 0 at the left-most pixel.

top - The top side position. Starting from 0 at the top-most scan line.

width - Number of pixels to be filled in each scan line.

height - Number of scan lines to be filled.

red - A number from 0 to 255, identifying the red intensity

green - A number from 0 to 255, identifying the green intensity

blue - A number from 0 to 255, identifying the blue intensity

alpha - (Only if the alpha channeled is used when bitmap created) A number from 0 to 255, identifying the alpha value.

Return value:

None

Comments:

This function set the color and (optionally) alpha value in specified region of the bitmap. NOTE: If alpha channel is used, this function does NOT composite the background with the source color, instead the background will be replaced by the source color and alpha. If alpha channel is not used, the "alpha" parameter is ignored.

FPDFBitmap_GetBuffer

Get data buffer of an FXDIB

Prototype:

void* FPDFBitmap_GetBuffer(FPDF_BITMAP bitmap);

Parameters:

bitmap - Handle to the bitmap. Returned by FPDFBitmap_Create function.

Return value:

The pointer to the first byte of the bitmap buffer.

Comments:

Applications can use this function to get the bitmap buffer pointer, then manipulate any color and/or alpha values for any pixels in the bitmap.

FPDFBitmap_GetWidth

Get width of an FXDIB

Prototype:

int FPDFBitmap_GetWidth(FPDF_BITMAP bitmap);

Parameters:

bitmap - Handle to the bitmap. Returned by FPDFBitmap_Create function.

Return value:

The number of pixels in a horizontal line of the bitmap.

FPDFBitmap_GetHeight

Get height of an FXDIB

Prototype:

int FPDFBitmap_GetHeight(FPDF_BITMAP bitmap);

Parameters:

bitmap - Handle to the bitmap. Returned by FPDFBitmap_Create function.

Return value:

The number of pixels in a vertical line of the bitmap.

FPDFBitmap_GetStride

Get number of bytes for each scan line in the bitmap buffer

Prototype:

int FPDFBitmap_GetStride(FPDF_BITMAP bitmap);

Parameters:

bitmap - Handle to the bitmap. Returned by FPDFBitmap_Create function.

Return value:

The number of bytes for each scan line in the bitmap buffer.

FPDFBitmap_Destroy

Destroy an FXDIB and release all related buffers

Prototype:

void FPDFBitmap_Destroy(FPDF_BITMAP bitmap);

Parameters:

bitmap - Handle to the bitmap. Returned by FPDFBitmap_Create function.

Return value:

None.

Comments:

This function will not destroy any external buffer.

FPDF_AllocMemory

Allocated memory block in FPDFSDK. This memory can be freed by FPDF_FreeMemory function.

Prototype:

void* FPDF_AllocMemory(unsigned long size);

Parameters:

size - Byte size of requested memory block. Can not be zero.

Return value:

The allocated pointer. NULL if memory not available.

Comments:

Some FPDFSDK interface may require application to allocate memory for internal use of FPDFSDK. In this case application must call this function to allocate memory, don't use malloc() or other memory allocator. If an error handler installed and exception/long jump is used in the out of memory handling, this function might never return if no memory available.

FPDF_FreeMemory

Free a memory area allocated by Foxit SDK.

Prototype:

void FPDF_FreeMemory(void* p);

Parameters:

p - The pointer. Should not be NULL.

Return value:

None.

Comments:

In case FPDFSDK allocated some memory for user application, the user application must free it to avoid memory leakage. And the application must call FPDF_FreeMemory function to do that. Do NOT use c/c++ memory free() function or other similar functions.

FPDF_SetErrorHandler

Set a call back function when FPDFSDK has some error to report

Prototype:

void FPDF_SetErrorHandler(FPDF_ErrorHandler func);

Parameters:

func - Pointer to the error handler function

Return value:

None.

Comments:

Currently only two error codes are defined (see above).

FPDF_SetModulePath

Set the folder path for module files (like the FPDFCJK.BIN).

Prototype:

void FPDF_SetModulePath(FPDF_STRING module_name, FPDF_STRING folder_name);

Parameters:

module_name - Name of the module. Currently please use NULL (0) only.

folder_name - Name of the folder. For example: "C:\\program files\\FPDFSDK".

Return value:

None.

FPDF_SetGlyphProvider

Make use of a custom glyph bitmap provider.

Not available on Desktop Windows system.

Prototype:

void FPDF_SetGlyphProvider(FPDF_GLYPHPROVIDER* pProvider);

Parameters:

pProvider - Pointer to a provider structure. This structure must be available all the time (better put it in static data). And all member interfaces of this structure should be properly set and implemented.

Return value:

None.

FPDF_SetSystemFontFile

Make use of a system font. The font file must be in TrueType or Type1 format and must be encoded in a standard encoding system. Available for embedded Linux system only.

Prototype:

int FPDF_SetSystemFontFile(FPDF_BYTESTRING file_path);

Parameters:

file_path - The full path of the font file.

Return value:

Non-zero for success. Zero for error.

FPDF_SetWCEFontMapper

For Windows Mobile Only: make use of a font mapper for CJK charsets. This function should be called before page rendering.

Prototype:

void FPDF_SetWCEFontMapper(FPDF_WCEFONTMAPPER* mapper);

Parameters:

mapper - Pointer to the mapper structure.

Return value:

None.

2. FPDFDOC Operation (Base Module)

FPDFBookmark_GetFirstChild

Get first child of a bookmark item, or first top level bookmark item.

Prototype:

FPDF_BOOKMARK FPDFBookmark_GetFirstChild(FPDF_DOCUMENT document, FPDF_BOOKMARK bookmark);

Parameters:

document - Handle to the document. Returned by FPDF_LoadDocument or FPDF_LoadMemDocument.

bookmark - Handle to the current bookmark. Can be NULL if you want to get the first top level item.

Return value:

Handle to the first child or top level bookmark item. NULL if no child or top level bookmark found.

FPDFBookmark_GetNextSibling

Get next bookmark item on the same level.

Prototype:

FPDF_BOOKMARK FPDFBookmark_GetNextSibling(FPDF_DOCUMENT document, FPDF_BOOKMARK bookmark);

Parameters:

document - Handle to the document. Returned by FPDF_LoadDocument or FPDF_LoadMemDocument.

bookmark - Handle to the current bookmark. Cannot be NULL.

Return value:

Handle to the next bookmark item on the same level. NULL if this is the last bookmark on this level.

FPDFBookmark_Find

Find a bookmark in the document, using the bookmark title

Prototype:

FPDF_BOOKMARK FPDFBookmark_Find(FPDF_DOCUMENT document, FPDF_WIDESTRING title);

Parameters:

document - Handle to the document. Returned by FPDF_LoadDocument or FPDF_LoadMemDocument

title - The UTF-16LE encoded Unicode string for the bookmark title to be searched. Can't be NULL.

Return value:

Handle to the found bookmark item. NULL if the title can't be found.

Comments:

It always returns the first found bookmark if more than one bookmarks have the same title.

FPDFBookmark_GetColorRef

Get designated color of a bookmark item

Prototype:

unsigned long FPDFBookmark_GetColorRef(FPDF_BOOKMARK bookmark);

Parameters:

bookmark - Handle to the bookmark

Return value:

A COLORREF value (0x00ggbbrr) for the bookmark item.

FPDFBookmark_GetFontStyle

Get designated font style for a bookmark item

Prototype:

unsigned long FPDFBookmark_GetFontStyle(FPDF_BOOKMARK bookmark);

Parameters:

bookmark - Handle to the bookmark

Return value:

A number indicating the font style, as defined below

0 Normal

1 Italic

2 Bold
3 Bolditalic

FPDFBookmark_GetTitle

Get title of a bookmark

Prototype:

```
unsigned long FPDFBookmark_GetTitle(FPDF_BOOKMARK bookmark,
                                    void* buffer, unsigned long buflen);
```

Parameters:

bookmark - Handle to the bookmark
buffer - A buffer for output the title. Can be NULL.
buflen - The length of the buffer, number of bytes. Can be 0.

Return value:

Number of bytes the title consumes, including trailing zeroes.

Comments:

No matter on what platform, the title is always output in UTF-16LE encoding, which means the buffer can be regarded as an array of WORD (on Intel and compatible CPUs), each WORD represent the Unicode of a character (some special Unicode may take 2 WORDs). The string is followed by two bytes of zero indicating end of the string. The return value always indicated number of bytes required for the buffer, even when there is no buffer specified, or the buffer size is less then required. In this case, the buffer will not be modified.

FPDFBookmark_GetAction

Get the action associated with a bookmark item

Prototype:

```
FPDF_ACTION FPDFBookmark_GetAction(FPDF_BOOKMARK bookmark);
```

Parameters:

bookmark - Handle to the bookmark

Return value:

Handle to the action data. NULL if no action is associated with this bookmark. In this case, the application should try FPDFBookmark_GetDest

FPDFBookmark_GetDest

Get the destination associated with a bookmark item

Prototype:

```
FPDF_DEST FPDFBookmark_GetDest(FPDF_DOCUMENT document,
                               FPDF_BOOKMARK bookmark);
```

Parameters:

document - Handle to the document

bookmark - Handle to the bookmark

Return value:

Handle to the destination data. NULL if no destination is associated with this bookmark

FPDFBookmark_GetPageFirstLine

Get first text line of the page, which can be used for alternative bookmark, if no bookmark is available

Prototype:

Int FPDFBookmark_GetPageFirstLine(FPDF_PAGE page, void* buffer, int buflen);

Parameters:

page - The page handle

buffer - A buffer for output the title. Can be NULL.

buflen - The length of the buffer, number of bytes. Can be 0.

Return value:

Number of bytes the text line consumes, including trailing zeroes.

Comments:

No matter on what platform, the title is always output in UTF-16LE encoding, which means the buffer can be regarded as an array of WORD (on Intel and compatible CPUs), each WORD represent the Unicode of a character (some special Unicode may take 2 WORDs). The string is followed by two bytes of zero indicating end of the string. The return value always indicated number of bytes required for the buffer, even when there is no buffer specified, or the buffer size is less then required. In this case, the buffer will not be modified.

Note: This is a temporary function. Itwill be deleted soon.

FPDFAction_GetType

Get type of an action

Prototype:

unsigned long FPDFAction_GetType(FPDF_ACTION action);

Parameters:

action - Handle to the action

Return value:

A type number as defined below

0	Unsupported action type.
1	Go to a destination within current document.
2	Go to a destination within another document.
3	Universal Resource Identifier, including web pages and other Internet based resources.
4	Launch an application or open a file.

FPDFAction_GetFilePath

Get file path of an remote goto action

Prototype:

unsigned long FPDFAction_GetFilePath(FPDF_ACTION action,
void* buffer, unsigned long buflen);

Parameters:

action - Handle to the action. Must be a RMEOTEGOTO or LAUNCH action
buffer - A buffer for output the path string. Can be NULL.
buflen - The length of the buffer, number of bytes. Can be 0.

Return value:

Number of bytes the file path consumes, including trailing zero.

Comments:

The file path is output in local encoding. The return value always indicated number of bytes required for the buffer, even when there is no buffer specified, or the buffer size is less then required. In this case, the buffer will not be modified.

FPDFAction_GetDest

Get destination of an action

Prototype:

FPDF_DEST FPDFAction_GetDest(FPDF_DOCUMENT document, FPDF_ACTION action);

Parameters:

document - Handle to the document.
action - Handle to the action. It must be a GOTO or REMOTEGOTO action.

Return value:

Handle to the destination data.

Comments:

In case of remote goto action, the application should first use FPDFAction_GetFilePath to get file path, then load that particular document, and use its document handle to call this function.

FPDFAction_GetURIPath

Get URI path of a URI action

Prototype:

unsigned long FPDFAction_GetURIPath(FPDF_DOCUMENT document, FPDF_ACTION action, void* buffer, unsigned long buflen);

Parameters:

document - Handle to the document.
action - Handle to the action. Must be a URI action
buffer - A buffer for output the path string. Can be NULL.

buflen - The length of the buffer, number of bytes. Can be 0.

Return value:

Number of bytes the URI path consumes, including trailing zero.

Comments:

The URI path is always encoded in 7-bit ASCII.

The return value always indicated number of bytes required for the buffer, even when there is no buffer specified, or the buffer size is less then required. In this case, the buffer will not be modified.

FPDFDest_GetPageIndex

Get page index of a destination

Prototype:

unsigned long FPDFDest_GetPageIndex(FPDF_DOCUMENT document, FPDF_DEST dest);

Parameters:

document - Handle to the document

dest - Handle to the destination

Return value:

The page index. Starting from 0 for the first page.

FPDFDest_GetZoomMode

Get the designated zoom mode of a destination

Prototype:

unsigned long FPDFDest_GetZoomMode(FPDF_DEST dest);

Parameters:

dest - Handle to the destination

Return value:

The zoom mode as defined below

1	Zoom level with specified offset.
2	Fit both the width and height of the page (whichever smaller).
3	Fit the page width.
4	Fit the page height.
5	Fit a specific rectangle area within the window.

FPDFDest_GetZoomParam

Get zoom parameters.

Prototype:

double FPDFDest_GetZoomParam(FPDF_DEST dest, int param);

Parameters:

dest - Handle to the destination.

param - Index of the parameter, starting with zero (see comments below).

Return value:

A float number for the zoom parameter.

Comments:

Different zoom mode has different parameters. Here is a list:

XYZ	Three parameters: x, y position in the page and the zoom ratio (0 for not specified).
FITPAGE	No parameters;
FITHORZ	One parameter: the top margin of the page.
FITVERT	One parameter: the left margin of the page.
FITRECT	Four parameters: the left, top, right, bottom margin of the fitting rectangle. Use 0-3 as parameter index for them, respectively.

FPDFLink_GetLinkAtPoint

Find a link at specified point on a document page

Prototype:

FPDF_LINK FPDFLink_GetLinkAtPoint(FPDF_PAGE page, double x, double y);

Parameters:

page - Handle to the document page

x - The x coordinate of the point, specified in page coordinate system.

y - The y coordinate of the point, specified in page coordinate system.

Return value:

Handle to the link. NULL if there's no link at that point.

Comments:

The point coordinate are specified in page coordinate system. You can convert coordinate from screen system to page system using FPDF_DeviceToPage functions.

FPDFLink_GetDest

Get destination info of a link.

Prototype:

FPDF_DEST FPDFLink_GetDest(FPDF_DOCUMENT document, FPDF_LINK link);

Parameters:

document - Handle to the document

link - Handle to the link. Returned by FPDFLink_GetLinkAtPoint

Return value:

Handle to the destination. NULL if there is no destination associated with the link, in this case the application should try FPDFLink_GetAction

FPDFLink_GetAction

Get action info of a link

Prototype:

FPDF_ACTION FPDFLink_GetAction(FPDF_LINK link);

Parameters:

link - Handle to the link.

Return value:

Handle to the action. NULL if there is no action associated with the link.

FPDF_ GetPageThumbnail

Get stored thumbnail image of a page. Only supported on Windows system.

Prototype:

HBITMAP FPDF_GetPageThumbnail(FPDF_PAGE page);

Parameters:

page - Handle to a page.

Return value:

Handle to a Windows bitmap storing the thumbnail. NULL if no thumbnail is stored for this page.

Comments:

If no thumbnail is stored for a page, the application can render the page into a small bitmap for thumbnail. The application should free the image with DeleteObject WIN32 function, when it's done with the thumbnail.

NOTE: This function is not supported on Windows CE.

FPDF_GetMetaText

Get a text from metadata of the document. Result is encoded in UTF-16LE.

Prototype:

unsigned long FPDF_GetMetaText(FPDF_DOCUMENT doc, FPDF_BYTESTRING tag,
void* buffer, unsigned long buflen);

Parameters:

doc - A document handle

tag - The tag for the meta data. Currently, It can be "Title", "Author", "Subject", "Keywords", "Creator", "Producer", "CreationDate", or "ModDate". For detailed explanation of these tags and their respective values, please refer to PDF Reference 1.6, section 10.2.1, "Document Information Dictionary".

buffer - A buffer for output the title. Can be NULL.

buflen - The length of the buffer, number of bytes. Can be 0.

Return value:

Number of bytes the title consumes, including trailing zeros.

Comments:

No matter on what platform, the title is always outputted in UTF-16LE encoding, which means the buffer can be regarded as an array of WORD (on Intel and compatible CPUs), each WORD represents the Unicode of a character (some special Unicode may take 2 WORDs). The string is followed by two bytes of zero indicating the end of the string. The return value always indicates number of bytes required for the buffer, even when there is no buffer specified, or the buffer size is less than required. In this case, the buffer will not be modified.

3. FPDFTEXT Operation (Text Module)

FPDFText_LoadPage

Prepare information about all characters in a page.

Prototype:

FPDF_TEXTPAGE FPDFText_LoadPage(FPDF_PAGE page);

Parameters:

page - Handle to the page. Returned by FPDF_LoadPage function

Return value:

A handle to the text page information structure.
NULL if something goes wrong.

Comments:

Application must call FPDFText_ClosePage to release the text page information.
If you didn't purchase Text Module , this function will return NULL.

FPDFText_ClosePage

Release all resources allocated for a text page information structure

Prototype:

void FPDFText_ClosePage(FPDF_TEXTPAGE text_page);

Parameters:

text_page - Handle to a text page information structure.
Returned by FPDFText_LoadPage function.

Return Value:

None.

FPDFText_CountChars

Get number of characters in a page

Prototype:

Int FPDFText_CountChars(FPDF_TEXTPAGE text_page);

Parameters:

text_page - Handle to a text page information structure.

Returned by FPDFText_LoadPage function.

Return value:

Number of characters in the page. Return -1 for error.
Generated characters, like additional space characters, new line characters, are also counted.

Comments:

Characters in a page form a "stream", inside the stream, each character has an index.
We will use the index parameters in many of FPDFTEXT functions. The first character in the page has an index value of zero.

FPDFText_GetUnicode

Get Unicode of a character in a page

Prototype:

unsigned int FPDFText_GetUnicode(FPDF_TEXTPAGE text_page, int index);

Parameters:

text_page - Handle to a text page information structure.
Returned by FPDFText_LoadPage function.
index - Zero-based index of the character

Return value:

The Unicode of the particular character.
If a character is not encoded in Unicode and Foxit engine can't convert to Unicode, the return value will be zero.

FPDFText_IsGenerated

Indicates whether a character is a generated character

Prototype:

FPDF_BOOL FPDFText_IsGenerated(FPDF_TEXTPAGE text_page, int index);

Parameters:

text_page - Handle to a text page information structure.
Returned by FPDFText_LoadPage function.
index - Zero-based index of the character

Return value:

TRUE indicates a generated character and FALSE indicates an actual character in the PDF page.

Comments:

"Generated character" is character not actually encoded in the PDF page, but generated by FPDFTEXT engine to keep formatting information.
This happens in two cases:

1) an extra space character will be generated if two characters in the same line appears to be apart quite some space

2) a new line character will be generated if two consecutive characters appears to be on different line. This characters are useful when doing the search.

FPDFText_GetFontSize

Get the font size of a particular character

Prototype:

double FPDFText_GetFontSize(FPDF_TEXTPAGE text_page, int index);

Parameters:

text_page - Handle to a text page information structure.
Returned by FPDFText_LoadPage function.

index - Zero-based index of the character.

Return value:

The font size of the particular character, measured in points (about 1/72 inch).
This is the typographic size of the font (so called "em size").

FPDFText_GetOrigin

Get origin position of a particular character

Prototype:

Void FPDFText_GetOrigin(FPDF_TEXTPAGE text_page, int index, double* x, double* y);

Parameters:

text_page - Handle to a text page information structure.
Returned by FPDFText_LoadPage function.

index - Zero-based index of the character.

x - Pointer to a double number receiving X position of the character origin.

y - Pointer to a double number receiving Y position of the character origin.

Return Value:

None.

Comments:

Origin X/Y positions are measured in PDF "user space".

FPDFText_GetCharBox

Get bounding box of a particular character

Prototype:

void FPDFText_GetCharBox(FPDF_TEXTPAGE text_page, int index, double* left,
double* right, double* bottom, double* top);

Parameters:

text_page - Handle to a text page information structure.
Returned by FPDFText_LoadPage function.

index - Zero-based index of the character.

left - Pointer to a double number receiving left position of the character box.
right - Pointer to a double number receiving right position of the character box.
bottom - Pointer to a double number receiving bottom position of the character box.
top - Pointer to a double number receiving top position of the character box.

Return Value:

None.

Comments:

All positions are measured in PDF "user space".

FPDFText_GetMatrix

Get the matrix of a particular character.

Prototype:

void FPDFText_GetMatrix(FPDF_TEXTPAGE text_page, int index, double* a, double* b, double* c, double* d);

Parameters:

text_page - Handle to a text page information structure.
Returned by FPDFText_LoadPage function.
index - Zero-based index of the character
a - Pointer to a double value receiving coefficient "a" of the matrix
b - Pointer to a double value receiving coefficient "b" of the matrix
c - Pointer to a double value receiving coefficient "c" of the matrix
d - Pointer to a double value receiving coefficient "d" of the matrix

Return value:

None.

Comments:

A matrix defines transformation of coordinate from one space to another.
In PDF, a matrix is defined by the following equations:
x' = a * x + c * y + e;
y' = b * x + d * y + f;
FPDFText_GetMatrix function is used to get a,b,c,d coefficients of the transformation from "text space" to "user space". The e, f coefficients are actually the origin position, which can be fetched by FPDFText_GetOrigin function.

FPDFText_GetFont

Get font of a particular character

Prototype:

FPDF_FONT FPDFText_GetFont(FPDF_TEXTPAGE text_page, int index);

Parameters:

text_page - Handle to a text page information structure.
Returned by FPDFText_LoadPage function.

index - Zero-based index of the character.

Return value:

A handle to the font used by the particular character.

This handle can be used in FPDFFont_xxx functions for more information about the font.

FPDFFont_GetAscent

Get font ascent (in 1/1000 em)

Prototype:

int FPDFFont_GetAscent(FPDF_FONT font);

Parameters:

font - Handle to a font. Returned by FPDFText_GetFont function.

Return value:

The ascent (typically the above-baseline height of letter "h"), measured in 1/1000 of em size. So if a character uses a font size (em size) of 10 points, and it has an ascent value of 500 (meaning half of the em), then the ascent height will be 5 points (5/72 inch).

FPDFFont_GetDescent

Get font descent (in 1/1000 em)

Prototype:

int FPDFFont_GetDescent(FPDF_FONT font);

Parameters:

font - Handle to a font. Returned by FPDFText_GetFont function.

Return value:

The descent (typically the under-baseline height of letter "g"), measured in 1/1000 of em size. Most fonts have a negative descent value.

FPDFFont_GetName

Get the Name of a font.

Prototype:

FPDF_BYTESTRING FPDFFont_GetName(FPDF_FONT font);

Parameters:

font - Handle to a font. Returned by FPDFText_GetFont function.

Return value:

A pointer to a null-terminated string that specifies the name of the font.

Application can't modify the returned string.

FPDFText_GetCharIndexAtPos

Get the index of a character at or nearby a certain position on the page

Prototype:

Int FPDFText_GetCharIndexAtPos(FPDF_TEXTPAGE text_page,
double x, double y, double xTorelance, double yTolerance);

Parameters:

text_page - Handle to a text page information structure.
Returned by FPDFText_LoadPage function.
x - X position in PDF "user space".
y - Y position in PDF "user space".
xTolerance - An x-axis tolerance value for character hit detection, in point unit.
yTolerance - A y-axis tolerance value for character hit detection, in point unit.

Return Value:

The zero-based index of the character at, or nearby the point (x,y).
If there is no character at or nearby the point, return value will be -1.
If an error occurs, -3 will be returned.

FPDFText_GetCharIndexByDirection

Move the character index in different directions and get new character index, from a specific character.

Prototype:

int FPDFText_GetCharIndexByDirection(FPDF_TEXTPAGE text_page,
int index, int direction);

Parameters:

text_page - Handle to a text page information structure.
Returned by FPDFText_LoadPage function.
index - Zero-based index for the current character
direction - A number indicating the moving direction. Can be one of the followings:
FPDFTEXT_LEFT,
FPDFTEXT_UP,
FPDFTEXT_RIGHT,
FPDFTEXT_DOWN

Return Value:

Zero-base character index for the new position.
-1 if beginning of the page reached; -2 if end of the page reached.
-3 for failures.

Comments:

FPDFTEXT moves the character pointer according to "stream order". For example, left will move to the previous character, right will move to next character. Because in PDF, "stream order" can be different from "appearance order" (the order that appears to human eyes), so it's possible the moving direction doesn't match the actually position movement. For example, using FPDFTEXT_LEFT may actually result in a character that's all the way down in the page.

FPDFText_GetText

Extract unicode text string from the page

Prototype:

int FPDFText_GetText(FPDF_TEXTPAGE text_page, int start_index,
int count, unsigned short* result);

Parameters:

text_page	-	Handle to a text page information structure. Returned by FPDFText_LoadPage function.
start_index	-	Index for the start characters.
count	-	Number of characters to be extracted.
result	-	A buffer (allocated by application) receiving the extracted unicodes. The size of the buffer must be able to hold the number of characters plus a terminator.

Return Value:

Number of characters written into the result buffer, excluding the trailing terminator.

Comments:

This function ignores characters without unicode information.

FPDFText_CountRects

Count number of rectangular areas occupied by a segment of texts.

Prototype:

int FPDFText_CountRects(FPDF_TEXTPAGE text_page, int start_index, int count);

Parameters:

text_page	-	Handle to a text page information structure. Returned by FPDFText_LoadPage function.
start_index	-	Index for the start characters
count	-	Number of characters

Return value:

Number of rectangles. Zero for error.

Comments:

This function, along with FPDFText_GetRect can be used by applications to detect the position on the page for a text segment, so proper areas can be highlighted or something. FPDFTEXT will automatically merge small character boxes into bigger one if those characters are on the same line and use same font settings.

FPDFText_GetRect

Get a rectangular area from the result generated by FPDFText_CountRects.

Prototype:

void FPDFText_GetRect(FPDF_TEXTPAGE text_page, int rect_index,
double* left, double* top, double* right, double* bottom);

Parameters:

text_page	-	Handle to a text page information structure. Returned by FPDFText_LoadPage function.
rect_index	-	Zero-based index for the rectangle.
left	-	Pointer to a double value receiving the rectangle left boundary.
top	-	Pointer to a double value receiving the rectangle top boundary.
right	-	Pointer to a double value receiving the rectangle right boundary.
bottom	-	Pointer to a double value receiving the rectangle bottom boundary.

Return Value:

None.

FPDFText_GetBoundedText

Extract unicode text within a rectangular boundary on the page

Prototype:

int FPDFText_GetBoundedText(FPDF_TEXTPAGE text_page,double left, double top, double right, double bottom,unsigned short* buffer,int buflen);

Parameters:

text_page	-	Handle to a text page information structure. Returned by FPDFText_LoadPage function.
left	-	Left boundary.
top	-	Top boundary.
right	-	Right boundary.
bottom	-	Bottom boundary.
buffer	-	A unicode buffer.
buflen	-	Number of characters (not bytes) for the buffer, excluding an additional terminator

Return Value:

If buffer is NULL or buflen is zero, number of characters (not bytes) needed, otherwise, number of characters copied into the buffer.

FPDFText_CountBoundedSegments

Get number of text segments within a rectangular boundary on the page

Prototype:

int FPDFText_CountBoundedSegments(FPDF_TEXTPAGE text_page,
double left, double top, double right, double bottom);

Parameters:

text_page	-	Handle to a text page information structure. Returned by FPDFText_LoadPage function.

left - Left boundary.
top - Top boundary.
right - Right boundary.
bottom - Bottom boundary.

Return Value:

Number of segments.

FPDFText_GetBoundedSegment

Get a particular segment in the result generated by FPDFText_CountBoundedSegments function.

Prototype:

```
Void FPDFText_GetBoundedSegment(FPDF_TEXTPAGE text_page, int seg_index,
                                int* start_index, int* count);
```

Parameters:

text_page - Handle to a text page information structure.
Returned by FPDFText_LoadPage function.
seg_index - Zero-based index for the segment
start_index - Pointer to an integer receiving the start character index for the segment.
count - Pointer to an integer receiving number of characters in the segment.

Return Value:

None.

FPDFText_FindStart

Start a search.

Prototype:

```
FPDF_SCHHANDLE FPDFText_FindStart(FPDF_TEXTPAGE text_page,
                                  FPDF_WIDESTRING findwhat, unsigned long flags, int start_index);
```

Parameters:

text_page - Handle to a text page information structure.
Returned by FPDFText_LoadPage function.
Findwhat - A unicode match pattern.
flags - Option flags.
start_index - Start from this character. -1 for end of the page.

Return Value:

A handle for the search context. FPDFText_FindClose must be called to release this handle.

FPDFText_FindNext

Search in the direction from page start to end.

Prototype:

FPDF_BOOL FPDFText_FindNext(FPDF_SCHHANDLE handle);

Parameters:

handle - A search context handle returned by FPDFText_FindStart.

Return Value:

Whether a match is found.

FPDFText_FindPrev

Search in the direction from page end to start.

Prototype:

FPDF_BOOL FPDFText_FindPrev(FPDF_SCHHANDLE handle);

Parameters:

handle - A search context handle returned by FPDFText_FindStart.

Return Value:

Whether a match is found.

FPDFText_GetSchResultIndex

Get the starting character index of the search result.

Prototype:

int FPDFText_GetSchResultIndex(FPDF_SCHHANDLE handle);

Parameters:

handle - A search context handle returned by FPDFText_FindStart.

Return Value:

Index for the starting character.

FPDFText_GetSchCount

Get the number of matched characters in the search result.

Prototype:

int FPDFText_GetSchCount(FPDF_SCHHANDLE handle);

Parameters:

handle - A search context handle returned by FPDFText_FindStart.

Return Value:

Number of matched characters.

FPDFText_FindClose

Release a search context.

Prototype:

void FPDFText_FindClose(FPDF_SCHHANDLE handle);

Parameters:

handle - A search context handle returned by FPDFText_FindStart.

Return Value:

None.

FPDFLink_LoadWebLinks

Prepare information about weblinks in a page.

Prototype:

FPDF_PAGELINK FPDFLink_LoadWebLinks(FPDF_TEXTPAGE text_page);

Parameters:

text_page - Handle to a text page information structure.
Returned by FPDFText_LoadPage function.

Return Value:

A handle to the page's links information structure.
NULL if something goes wrong.

Comments:

Weblinks are those links implicitly embedded in PDF pages. PDF also has a type of annotation called "link", FPDFTEXT doesn't deal with that kind of link. FPDFTEXT weblink feature is useful for automatically detecting links in the page contents. For example, things like "http://www.foxitsoftware.com" will be detected, so applications can allow user to click on those characters to activate the link, even the PDF doesn't come with link annotations. FPDFLink_CloseWebLinks must be called to release resources.

FPDFLink_CountWebLinks

Count number of detected web links.

Prototype:

int FPDFLink_CountWebLinks(FPDF_PAGELINK link_page);

Parameters:

link_page - Handle returned by FPDFLink_LoadWebLinks.

Return Value:

Number of detected web links.

FPDFLink_GetURL

Fetch the URL information for a detected web link.

Prototype:

int FPDFLink_GetURL(FPDF_PAGELINK link_page, int link_index,
unsigned short* buffer,int buflen);

Parameters:

link_page - Handle returned by FPDFLink_LoadWebLinks.
link_index - Zero-based index for the link.

buffer - A unicode buffer.
buflen - Number of characters (not bytes) for the buffer, excluding an additional terminator.

Return Value:

If buffer is NULL or buflen is zero, return number of characters (not bytes) needed, otherwise, return number of characters copied into the buffer.

FPDFLink_CountRects

Count number of rectangular areas for the link.

Prototype:

int FPDFLink_CountRects(FPDF_PAGELINK link_page, int link_index);

Parameters:

link_page - Handle returned by FPDFLink_LoadWebLinks.
link_index - Zero-based index for the link.

Return Value:

Number of rectangular areas for the link.

FPDFLink_GetRect

Fetch the boundaries of a rectangle for a link.

Prototype:

Void FPDFLink_GetRect(FPDF_PAGELINK link_page, int link_index, int rect_index, double* left, double* top,double* right, double* bottom);

Parameters:

link_page - Handle returned by FPDFLink_LoadWebLinks.
link_index - Zero-based index for the link.
rect_index - Zero-based index for a rectangle.
left - Pointer to a double value receiving the rectangle left boundary.
top - Pointer to a double value receiving the rectangle top boundary.
right - Pointer to a double value receiving the rectangle right boundary.
bottom - Pointer to a double value receiving the rectangle bottom boundary.

Return Value:

None.

FPDFLink_CloseWebLinks

Release resources used by weblink feature.

Prototype:

Void FPDFLink_CloseWebLinks(FPDF_PAGELINK link_page);

Parameters:

link_page - Handle returned by FPDFLink_LoadWebLinks.

Return Value:

None.

FPDFText_PDFToText

Convert a PDF file to a TXT File.

Prototype:

FPDF_BOOL FPDFText_PDFToText(const char * sour_file,const char * dest_file,
int flag,FPDF_BYTESTRING password);

Parameters:

sour_file - Path to the PDF file you want to convert.
dest_file - The path of the file you want to save.
flag - 0 for stream order ,1 for appearance order.
password - A string used as the password for PDF file.
If no password needed, empty or NULL can be used.

Return value:

TURE for succeed, False for failed.

FPDFText_PageToText

Convert a PDF page data to a text buffer.

Prototype:

int FPDFText_PageToText(FPDF_DOCUMENT doc,
int page_index,wchar_t* buf,int size,int flag);

Parameters:

doc - Handle to document. Returned by FPDF_LoadDocument function.
page_index - Index number of the page. 0 for the first page.
buf - An output buffer used to hold the text of the page.
size - Size of the buffer.
flag - 0 for stream order ,1 for appearance order.

Return value:

If buf is NULL or size is zero, number of characters (not bytes) needed,
otherwise, number of characters copied into the buf.

4. FPDFEDIT Operation (Edit Module)

a) Document Operation

FPDF_CreateNewDocument

Create a new PDF document.

Prototype:

FPDF_DOCUMENT FPDF_CreateNewDocument;

Parameters:

None.

Return value:

A handle to a document. If failed, NULL is returned.

FPDF_SaveAsFile

Save the specified document with a new name or format.

Prototype:

```
FPDF_BOOL    FPDF_SaveAsFile(FPDF_DOCUMENT document, char* file_name,
                             FPDF_DWORD permissions,
                             FPDF_BYTE const* UserPwd, long nUserPwdLen,
                             FPDF_BYTE const* OwnerPwd, long nOwnerPwdLen);
```

Parameters:

Document	-	Handle to document. Returned by FPDF_LoadDocument or FPDF_CreateNewDocument.
file_name	-	Path to the PDF file (including extension).
permissions	-	The PDF document permissions.
UserPwd	-	A 32-byte string, the user password to entered. Could be NULL.
nUserPwdLen	-	The length of the UserPwd.
OwnerPwd	-	A 32-byte string, the owner password to entered. Could be NULL.
nOwnerPwdLen	-	The length of the OwnerPwd.

Return value:

TRUE for succeed, FALSE for Failed.

b) Page Operation

FPDFPage_New

Construct an empty page.

Prototype

```
FPDF_PAGE    FPDFPage_New(FPDF_DOCUMENT document, int page_index,
                          double width, double height);
```

Parameters:

document	-	Handle to document. Returned by FPDF_LoadDocument or FPDF_CreateNewDocument.
page_index	-	The index of a page.
width	-	The page width.
height	-	The page height.

Return value:

The handle to a page.

Comments:

Loaded page can be deleted by FPDFPage_Delete.

FPDFPage_Delete

Delete a PDF page.

Prototype:

Void FPDFPage_Delete(FPDF_DOCUMENT document, int page_index);

Parameters:

document - Handle to document.
Returned by FPDF_LoadDocument or FPDF_CreateNewDocument.

page_index - The index of a page.

Return value:

None.

FPDFPage_GetRectangle

Get page rectangle, in points (1 point equals 1/72 inch).

Prototype:

FPDF_BOOL FPDFPage_GetRectangle(FPDF_PAGE page, int iRect,
double* left, double* right, double* bottom, double* top);

Parameters:

page - Handle to a page. Returned by FPDFPage_New.

iRect - The index values for getting boxes defined following.

FPDF_RECT_PAGE	0
FPDF_RECT_BOUNDING	1
FPDF_RECT_MEDIABOX	2
FPDF_RECT_CROPBOX	3
FPDF_RECT_TRIMBOX	4
FPDF_RECT_ARTBOX	5
FPDF_RECT_BLEEDBOX	6

left - Pointer to a double value receiving the left of a rectangle (in points).

right - Pointer to a double value receiving the right of a rectangle (in points).

bottom - Pointer to a double value receiving the bottom of a rectangle (in points).

top - Pointer to a double value receiving the top of a rectangle (in points).

Return value:

TRUE if successful, FALSE otherwise.

FPDFPage_SetRectangle

Set page rectangle, in points (1 point equals 1/72 inch)

Prototype:

FPDF_BOOL FPDFPage_SetRectangle(FPDF_PAGE page, int iRect,

double left, double right, double bottom, double top);

Parameters:

page - Handle to a page. Returned by FPDFPage_New.

iRect - The index values for getting boxes defined following.

FPDF_RECT_PAGE	0
FPDF_RECT_BOUNDING	1
FPDF_RECT_MEDIABOX	2
FPDF_RECT_CROPBOX	3
FPDF_RECT_TRIMBOX	4
FPDF_RECT_ARTBOX	5
FPDF_RECT_BLEEDBOX	6

left - The left coordinate of a rectangle (in points).

right - The right coordinate of a rectangle (in points).

bottom - The bottom coordinate of a rectangle (in points).

top - The top coordinate of a rectangle (in points).

Return value:

TRUE if successful, FALSE otherwise.

Comment:

When it's done, the content in this page will be changed.

You must call the FPDF_LoadPage to reload the page.

FPDFPage_GetRotation

Get the page rotation. One of following values will be returned：0(0), 1(90), 2(180), 3(270).

Prototype:

Int FPDFPage_GetRotation(FPDF_PAGE page);

Parameters:

page - Handle to a page. Returned by FPDFPage_New.

Return value:

The PDF page rotation.

Comment:

The PDF page rotates clockwise.

FPDFPage_SetRotation

Set page rotation. One of following values will be set: 0(0), 1(90), 2(180), 3(270).

Prototype:

Void FPDFPage_SetRotation(FPDF_PAGE page, int rotate);

Parameters:

page - Handle to a page. Returned by FPDFPage_New.

rotate - The value of the PDF page rotation

Return value:

None.

Comment:

The PDF page rotates clockwise. When it's done, the content in this pagewill be changed. You must call the FPDF_LoadPage to reload the page.

FPDFPage_InsertObject

Insert an object to the page. The page object is automatically freed.

Prototype:

void FPDFPage_InsertObject(FPDF_PAGE page, FPDF_PAGEOBJECT page_obj);

Parameters:

page - Handle to a page. Returned by FPDFPage_New.

page_obj - Handle to a page object. Returned by FPDFPageObj_NewTextObj FPDFPageObj_NewTextObjEx; FPDFPageObj_NewPathObj

Return value:

None.

FPDFPage_DeleteObject

Delete an object from the page. The page object is automatically freed.

Prototype:

Void FPDFPage_DeleteObject(FPDF_PAGE page, int index);

Parameters:

page - Handle to a page. Returned by FPDFPage_New.

index - The index of a page object.

Return value:

None.

FPDFPage_CountObject

Get number of page objects inside the page

Prototype:

int FPDFPage_CountObject(FPDF_PAGE page);

Parameters:

Page - Handle to a page. Returned by FPDFPage_New.

Return value:

The number of the page object.

FPDFPage_GetObject

Get page object by index.

Prototype:

FPDF_PAGEOBJECT FPDFPage_GetObject(FPDF_PAGE page, int index);

Parameters:

page - Handle to a page. Returned by FPDFPage_New.

index - The index of a page object.

Return value:

The handle of the page object. Null for failed.

FPDFPage_GetPageObjectIndex

Get the index of the page object in the specify page.

Prototype:

```
int FPDFPage_GetPageObjectIndex(FPDF_PAGE page,
                                FPDF_PAGEOBJECT page_obj);
```

Parameters:

page - Handle to a page. Returned by FPDFPage_New.

page_obj - Handle to a page object. Returned by FPDFPageObj_NewTextObj, FPDFPageObj_NewTextObjEx,FPDFPageObj_NewPathObj, etc.

Return value:

If successful, returns the index of the page object. Otherwise, returns -1.

FPDFPage_GenerateContent

Generate PDF Page content.

Prototype:

```
FPDF_BOOL FPDFPage_GenerateContent (FPDF_PAGE page);
```

Parameters:

page - Handle to a page. Returned by FPDFPage_New.

Return value:

True if successful, false otherwise.

Comment:

Before you save the page to a file, or reload the page, you must call the FPDFPage_GenerateContent function. Or the changed information will be lost.

c) Page Objects Operation

FPDFPageObj_Transform

Transform (scale, rotate, shear, move) page object.

Prototype:

```
void FPDFPageObj_Transform(FPDF_PAGEOBJECT page_object,
                           double a, double b, double c, double d, double e, double f);
```

Parameters:

page_object - Handle to a page object. Returned by FPDFPageObj_NewTextObj,

FPDFPageObj_NewTextObjEx,FPDFPageObj_NewPathObj, etc.

a - The coefficient "a" of the matrix.
b - The coefficient "b" of the matrix.
c - The coefficient "c" of the matrix.
d - The coefficient "d" of the matrix.
e - The coefficient "e" of the matrix.
f - The coefficient "f" of the matrix.

Return value:

None.

FPDFPageObj_Clone

Create a new page object based on this page object.

Prototype:

FPDF_PAGEOBJECT FPDFPageObj_Clone(FPDF_PAGEOBJECT page_object);

Parameters:

page_object - Handle to a page object. Returned by FPDFPageObj_NewTextObj, FPDFPageObj_NewTextObjEx,FPDFPageObj_NewPathObj, etc.

Return value:

Handle to a page object.

FPDFPageObj_Free

Free a page object.

Prototype:

Void FPDFPageObj_Free(FPDF_PAGEOBJECT page_object);

Parameters:

page_object - Handle to a page object. Returned by FPDFPageObj_NewTextObj, FPDFPageObj_NewTextObjEx,FPDFPageObj_NewPathObj,etc.

Return value:

None.

FPDFPageObj_GetType

Get page object type.

Prototype:

int FPDFPageObj_GetType(FPDF_PAGEOBJECT page_object);

Parameters:

page_object - Handle to a page object. Returned by FPDFPageObj_NewTextObj, FPDFPageObj_NewTextObjEx,FPDFPageObj_NewPathObj, etc.

Return value:

Returns one of the FPDF_PAGEOBJ_xxxx constants defined following.

FPDF_PAGEOBJ_TEXT	1
FPDF_PAGEOBJ_PATH	2
FPDF_PAGEOBJ_IMAGE	3
FPDF_PAGEOBJ_SHADING	4
FPDF_PAGEOBJ_FORM	5

FPDFPageObj_GetBBox

Get page object bounding box.

Prototype:

```
void    FPDFPageObj_GetBBox(FPDF_PAGEOBJECT page_object,
                            double* left, double* bottom, double* right, double* top );
```

Parameters:

page_object - Handle to a page object. Returned by FPDFPageObj_NewTextObj, FPDFPageObj_NewTextObjEx,FPDFPageObj_NewPathObj, etc.

left - Pointer to a double value receiving the left of a rectangle (in points).

bottom - Pointer to a double value receiving the bottom of a rectangle (in points).

right - Pointer to a double value receiving the right of a rectangle (in points).

top - Pointer to a double value receiving the top of a rectangle (in points).

Return value:

None.

FPDFPageObj_GetClipCount

Get the clip number of a page_objcet.

Prototype:

```
int  FPDFPageObj_GetClipCount(FPDF_PAGEOBJECT page_object);
```

Parameters:

page_object - Handle to a page object. Returned by FPDFPageObj_NewTextObj, FPDFPageObj_NewTextObjEx,FPDFPageObj_NewPathObj,etc.

Return value:

The clip count.

FPDFPageObj_GetClip

Get the clip path of a page object.

Prototype:

```
FPDF_PATH    FPDFPageObj_GetClip(FPDF_PAGEOBJECT page_object, int index);
```

Parameters:

page_object - Handle to a page object. Returned by FPDFPageObj_NewTextObj, FPDFPageObj_NewTextObjEx,FPDFPageObj_NewPathObj, etc.

index - The index of a path.

Return value:

The handle of a path.

FPDFPageObj_AddClip

Add a clip to a page object.

Prototype:

FPDF_BOOL FPDFPageObj_AddClip(FPDF_PAGEOBJECT page_object, FPDF_PATH path, int type);

Parameters:

page_object - Handle to a page object. Returned by FPDFPageObj_NewTextObj, FPDFPageObj_NewTextObjEx, FPDFPageObj_NewPathObj, etc.

path - Handle to a path. Returned by FPDFPathObj_GetPath.

type - The fill mode types defined following.

FPDF_FILL_NULL	0
FPDF_FILL_ALTERNATE	1
FPDF_FILL_WINDING	2

Return value:

True if successful, false otherwise.

FPDFPageObj_AppendPathToClip

Append a path to the current clip.

Prototype:

FPDF_BOOL FPDFPageObj_AppendPathToClip(FPDF_PAGEOBJECT page_object, FPDF_PATH path, int type);

Parameters:

page_object - Handle to a page object. Returned by FPDFPageObj_NewTextObj, FPDFPageObj_NewTextObjEx,FPDFPageObj_NewPathObj, etc.

path - Handle to a path. Returned by FPDFPathObj_GetPath.

type - The fill mode types defined following.

FPDF_FILL_NULL	0
FPDF_FILL_ALTERNATE	1
FPDF_FILL_WINDING	2

Return value:

True if successful, false otherwise.

FPDFPageObj_RemoveClip

Remove a clip from the page object.

Prototype:

void FPDFPageObj_RemoveClip(FPDF_PAGEOBJECT page_object, int index);

Parameters:

page_object - Handle to a page object. Returned by FPDFPageObj_NewTextObj, FPDFPageObj_NewTextObjEx, FPDFPageObj_NewPathObj, FPDFPageObj_NewImageObj etc.

index - The index of the clip.

Return value:

None.

FPDFPageObj_GetFillColor

Get the fill color of a page object.

Prototype:

FPDF_DWORD FPDFPageObj_GetFillColor(FPDF_PAGEOBJECT page_object);

Parameters:

page_object - Handle of page object, could be text_object, path_object and uncolored images.

Return value:

The fill color of a page object. Constructed by 0xaarrggbb.

FPDFPageObj_SetFillColor

Set the fill color of a page object.

Prototype:

void FPDFPageObj_SetFillColor(FPDF_PAGEOBJECT page_object, FPDF_DWORD argb);

Parameters:

page_object - Handle of page object, could be text_object, path_object and uncolored images.

argb - The fill color of a page object. Constructed by 0xaarrggbb.

Return value:

None.

FPDFPageObj_GetStrokeColor

Get the stroke color of a page object.

Prototype:

FPDF_DWORD FPDFPageObj_GetStrokeColor(FPDF_PAGEOBJECT page_object);

Parameters:

page_object - Handle of page object, could be text_object, path_object and uncolored images.

Return value:

The stroke color of a page object. Constructed by 0xaarrggbb.

FPDFPageObj_SetStrokeColor

Set the stroke color of a page object.

Prototype:

void FPDFPageObj_SetStrokeColor (FPDF_PAGEOBJECT page_object,
FPDF_DWORD argb);

Parameters:

page_object - Handle of page object, could be text_object, path_object and uncolored images.
argb - The stroke color of a page object. Constructed by 0xaarrggbb.

Return value:

None.

FPDFPageObj_GetLineWidth

Get the line width of a path_object.

Prototype:

double FPDFPageObj_GetLineWidth (FPDF_PAGEOBJECT page_object);

Parameters:

page_object - Handle of page object. It could be path_object or stroke text_object.

Return Value:

Return the line width of the page object.

FPDFPageObj_SetLineWidth

Set the line width of a path_object.

Prototype:

void FPDFPageObj_SetLineWidth (FPDF_PAGEOBJECT page_object, double width);

Parameters:

page_object - Handle of page object. It could be path_object or stroke text_object.
width - The line width of the page object.

Return Value:

None.

FPDFPageObj_GetLineCapStyle

Get the line cap style of a page object.

Prototype:

int FPDFPageObj_GetLineCapStyle (FPDF_PAGEOBJECT page_object);

Parameters:

page_object - Handle of page object. It could be path_object or stroke text_object.

Return Value:

The line cap styles defined following.

FPDF_LINECAP_BUTT 0

FPDF_LINECAP_ROUND 1
FPDF_LINECAP_PROJECT 2

FPDFPageObj_SetLineCapStyle

Set the line cap style of a page_object.

Prototype:

void FPDFPageObj_SetLineCapStyle (FPDF_PAGEOBJECT page_object, int style);

Parameters:

page_object - Handle of page object. It could be path_object or stroke text_object.

style - The line cap styles defined following.

FPDF_LINECAP_BUTT 0
FPDF_LINECAP_ROUND 1
FPDF_LINECAP_PROJECT 2

Return Value:

None.

FPDFPageObj_GetLineJoinStyle

Get the line join style of a page_object.

Prototype:

int FPDFPageObj_GetLineJoinStyle(FPDF_PAGEOBJECT page_object);

Parameters:

page_object - Handle of page object. It could be path_object or stroke text_object.

Return Value:

Return the line join styles defined following.

FPDF_LINEJOIN_MITER 0
FPDF_LINEJOIN_ROUND 1
FPDF_LINEJOIN_BEVEL 2

FPDFPageObj_SetLineJoinStyle

Set the line join style of a page object.

Prototype:

void FPDFPageObj_SetLineJoinStyle(FPDF_PAGEOBJECT page_object, int style);

Parameters:

page_object - Handle of page object. It could be path_object or stroke text_object.

Style - The line join styles defined following.

FPDF_LINEJOIN_MITER 0
FPDF_LINEJOIN_ROUND 1
FPDF_LINEJOIN_BEVEL 2

Return Value:

None.

FPDFPageObj_GetMiterLimit

Get the meter limit of a page object.

Prototype: FPDFFont_GetFlags

double FPDFPageObj_GetMiterLimit(FPDF_PAGEOBJECT page_object);

Parameters:

page_object - Handle of page object. It could be path_object or stroke text_object.

Return Value:

The meter limit of a page object.

Comments:

The miter limit imposes a maximum on the ratio of the miter length to the line width. When the limit is exceeded, the join will be converted from a miter to a bevel.

FPDFPageObj_SetMiterLimit

Set the meter limit of a page_object.

Prototype:

void FPDFPageObj_SetMiterLimit(FPDF_PAGEOBJECT page_object,
double miter_limit);

Parameters:

page_object - Handle of page object. It could be path_object or stroke text_object.

miter_limit - The meter limit of a page_object.

Return Value:

None.

Comments:

The miter limit imposes a maximum on the ratio of the miter length to the line width. When the limit is exceeded, the join is converted from a miter to a bevel.

FPDFPageObj_GetDashCount

Get dash count of a page object.

Prototype:

int FPDFPageObj_GetDashCount(FPDF_PAGEOBJECT page_object)

Parameters:

page_object - Handle of page object. It could be path_object or stroke text_object.

Return Value:

The dash count of a page object.

FPDFPageObj_SetDashCount

Set dash count of a page object.

Prototype:

Void FPDFPageObj_SetDashCount(FPDF_PAGEOBJECT page_object, int count);

Parameters:

page_object - Handle of page object. It could be path_object or stroke text_object.

count - The dash count of a page object.

Return Value:

None.

Comments:

Setting the count will release the old dash array of the page object and allocate a new dash array.

FPDFPageObj_GetDashArray

Get dash array elements of a page object.

Prototype:

Double FPDFPageObj_GetDashArray(FPDF_PAGEOBJECT page_object, int index);

Parameters:

page_object - Handle of page object. It could be path_object or stroke text_object.

index - The index of a dash array element.

Return Value:

Return the dash value of specific index of the dash array.

Comments:

The dash array's elements are numbers that specify the lengths of alternating dashes and gaps, the numbers must be nonnegative and not all zero.

FPDFPageObj_SetDashArray

Set dash array elements of a page object.

Prototype:

Void FPDFPageObj_SetDashArray(FPDF_PAGEOBJECT page_object, int index,
double dash_value);

Parameters:

page_object - Handle of page object. It could be path_object or stroke text_object.

index - The index of a dash array element.

dash_value - The dash value of specific index of the dash array.

Return Value:

None.

Comments:

The dash array's elements are numbers that specify the lengths of alternating dashes and gaps, the numbers must be nonnegative and not all zero.

FPDFPageObj_GetDashPhase

Get dash phase of a page object.

Prototype:

double FPDFPageObj_GetDashPhase(FPDF_PAGEOBJECT page_object);

Parameters:

page_object - Handle of page object. It could be path_object or stroke text_object.

Return Value:

The dash phase value of a page object.

Comments:

The dash phase specifies the distance into the dash pattern at where you can start the dash.

FPDFPageObj_SetDashPhase

Set dash phase of a page object.

Prototype:

void FPDFPageObj_SetDashPhase(FPDF_PAGEOBJECT page_object,
double phase_value);

Parameters:

page_object - Handle of path object. It could be path_object or stroke text_object.

phase_value - The dash phase value.

Return Value:

None.

Comments:

The dash phase specifies the distance into the dash pattern where you can start the dash.

d) Text Operation

FPDFPageObj_NewTextObj

Create a new Text Object.

Prototype:

FPDF_PAGEOBJECT FPDFPageObj_NewTextObj(void);

Parameters:

None.

Return Value:

Handle of text object.

FPDFPageObj_NewTextObjEx

Create a new Text Object.

Prototype:

FPDF_PAGEOBJECT FPDFPageObj_NewTextObjEx(unsigned short* text, int nwSize,
FPDF_FONT font);

Parameters:

text - Pointer to a string. The text you want to add in a text object.

nwSize - The length of the string.

font - Handle of FPDF_FONT. It could be FPDFTextObj_GetFont, FPDFFont_AddTrueType and FPDFFont_AddStandardFont.

Return Value:

Handle of text object.

FPDFTextObj_GetFont

Get the font handle of a text object.

Prototype:

```
FPDF_FONT FPDFTextObj_GetFont(FPDF_PAGEOBJECT text_object);
```

Parameters:

text_object - Handle of text object returned by FPDFPageObj_NewTextObj or FPDFPageObj_NewTextObjEx.

Return Value:

Handle of the font.

FPDFTextObj_SetFont

Set the font into a text object.

Prototype:

```
Void FPDFTextObj_SetFont(FPDF_PAGEOBJECT text_object, FPDF_FONT font,
    double font_size);
```

Parameters:

text_object - Handle of text object returned by FPDFPageObj_NewTextObj or FPDFPageObj_NewTextObjEx.

font - Handle of FPDF_FONT. It could be returned by FPDFTextObj_GetFont, FPDFFont_AddTrueType and FPDFFont_AddStandardFont.

font_size - The size of the font.

Return Value:

None.

FPDFTextObj_GetFontSize

Get the font size of a text object.

Prototype:

```
double FPDFTextObj_GetFontSize(FPDF_PAGEOBJECT text_object);
```

Parameters:

text_object - Handle of text object returned by FPDFPageObj_NewTextObj or FPDFPageObj_NewTextObjEx.

Return Value:

The value of the font size.

FPDFTextObj_GetMatrix

Get the text object matrix.

Prototype:

void FPDFTextObj_GetMatrix(FPDF_PAGEOBJECT text_object,
double* a, double* b, double* c, double* d, double* e, double* f);

Parameters:

text_object - Handle of text object returned by FPDFPageObj_NewTextObj or FPDFPageObj_NewTextObjEx.
a - Pointer to a double value receiving the coefficient "a" of the matrix.
b - Pointer to a double value receiving the coefficient "b" of the matrix.
c - Pointer to a double value receiving the coefficient "c" of the matrix.
d - Pointer to a double value receiving the coefficient "d" of the matrix.
e - Pointer to a double value receiving the coefficient "e" of the matrix.
f - Pointer to a double value receiving the coefficient "f" of the matrix.

Return Value:

None.

FPDFTextObj_SetMatrix

Set the text object matrix.

Prototype:

Void FPDFTextObj_SetMatrix(FPDF_PAGEOBJECT text_object,
double a, double b, double c, double d, double e, double f);

Parameters:

text_object - Handle of text object returned by FPDFPageObj_NewTextObj or FPDFPageObj_NewTextObjEx.
a - The coefficient "a" of the matrix.
b - The coefficient "b" of the matrix.
c - The coefficient "c" of the matrix.
d - The coefficient "d" of the matrix.
e - The coefficient "e" of the matrix.
f - The coefficient "f" of the matrix.

Return Value:

None.

FPDFTextObj_CountChars

Get the number of characters from a text object.

Prototype:

int FPDFTextObj_CountChars(FPDF_PAGEOBJECT text_object);

Parameters:

text_object - Handle of text object returned by FPDFPageObj_NewTextObj or FPDFPageObj_NewTextObjEx.

Return Value:

A character count in the text object.

FPDFTextObj_GetUincode

Get the unicode of a special character in a text object.

Prototype:

int FPDFTextObj_GetUincode(FPDF_PAGEOBJECT text_object, int index);

Parameters:

text_object - Handle of text object returned by FPDFPageObj_NewTextObj or FPDFPageObj_NewTextObjEx.

index - The index of the character to get the unicode.

Return Value:

The unicode value.

FPDFTextObj_SetUnicode

Set the unicode of a special character in a text object.

Protype::

FPDF_BOOL FPDFTextObj_SetUnicode(FPDF_PAGEOBJECT text_object, int index, int nUnicode);

Parameters:

text_object - Handle of text object returned by FPDFPageObj_NewTextObj or FPDFPageObj_NewTextObjEx.

index - The index of the character to set unicode.

nUnicode - The unicode value.

Return Value:

TRUE for successful, FALSE otherwise.

FPDFTextObj_GetOffset

Calculate the physical offset from the start to the specified special character.

Prototype:

double FPDFTextObj_GetOffset(FPDF_PAGEOBJECT text_object, int index);

Parameters:

text_object - Handle of text object returned by FPDFPageObj_NewTextObj or FPDFPageObj_NewTextObjEx.

index - The index of the character to calculate the offset to.

Return Value:

Return the offset value.

FPDFTextObj_GetTextMode

Get the text rendering mode.

Prototype:

int FPDFTextObj_GetTextMode(FPDF_PAGEOBJECT text_object);

Parameters:

text_object - Handle of text object returned by FPDFPageObj_NewTextObj or FPDFPageObj_NewTextObjEx.

Return Value:

Return one of the enumeration values defined in text-rendering modes.

FPDFTextObj_SetTextMode

Set the text rendering mode.

Prototype:

FPDF_BOOL FPDFTextObj_SetTextMode(FPDF_PAGEOBJECT text_object, int text_mode);

Parameters:

text_object - Handle of text object returned by FPDFPageObj_NewTextObj or FPDFPageObj_NewTextObjEx.

text_mode - One of the enumeration values defined in text rendering modes.

Return Value:

TRUE for successful, FALSE otherwise.

FPDFTextObj_GetCharSpace

Get the character-spacing value of a text object.

Prototype:

double FPDFTextObj_GetCharSpace(FPDF_PAGEOBJECT text_object);

Parameters:

text_object - Handle of text object returned by FPDFPageObj_NewTextObj or FPDFPageObj_NewTextObjEx.

Return Value:

The character-spacing value

Comment:

The character-spacing value is a number specified in unscaled text space units.

FPDFTextObj_SetCharSpace

Set the character-spacing value of a text object.

Prototype:

FPDF_BOOL FPDFTextObj_SetCharSpace(FPDF_PAGEOBJECT text_object,
double charspace);

Parameters:

text_object - Handle of text object returned by FPDFPageObj_NewTextObj or FPDFPageObj_NewTextObjEx.

charspace - The character-spacing value.

Return Value:

TRUE for successful, FALSE otherwise.

Comment:

The character-spacing value is a number specified in unscaled text space units.

FPDFTextObj_GetWordSpace

Get the word-spacing value of a text object.

Prototype:

double FPDFTextObj_GetWordSpace(FPDF_PAGEOBJECT text_object);

Parameters:

text_object - Handle of text object returned by FPDFPageObj_NewTextObj or FPDFPageObj_NewTextObjEx.

Return Value:

The word-spacing value.

Comment:

The word-spacing value is a number expressed in unscaled text space units.

FPDFTextObj_SetWordSpace

Set the word-spacing value of a text object.

Prototype:

FPDF_BOOL FPDFTextObj_SetWordSpace(FPDF_PAGEOBJECT text_object,
double wordspace);

Parameters:

text_object - Handle of text object returned by FPDFPageObj_NewTextObj or FPDFPageObj_NewTextObjEx.

wordspace - The word-spacing value.

Return Value:

TRUE if successful, FALSE otherwise.

FPDFTextObj_Insert

Inserts a substring at the given index within the text object.

Prototype:

int FPDFTextObj_Insert(FPDF_PAGEOBJECT text_object, unsigned short* text, int nwSize, int index);

Parameters:

text_object - Handle of text object returned by FPDFPageObj_NewTextObj or FPDFPageObj_NewTextObjEx.

text - A pointer to the substring to be inserted.

nwSize - The length of the substring.

index - The index of the character before which the insertion will take place.

Return Value:

Return the length of the changed string.

Comment:

The index can be 0 and the number of characters in a text object (returned by FPDFTextObj_CountChars).

FPDFTextObj_Delete

Delete a character or characters in the text object.

Prototype:

int FPDFTextObj_Delete(FPDF_PAGEOBJECT text_object, int index, int nCount = 1);

Parameters:

text_object - Handle of text object returned by FPDFPageObj_NewTextObj or FPDFPageObj_NewTextObjEx.

index - The index of the first character to delete.

nCount - The number of characters to be removed.

Return Value:

Return the characters number of the changed text object.

Comment:

Call this member function to delete a character or characters from a string starting with the character at index. If nCount is longer than the string, the remainder of the string will be removed.

FPDFTextObj_Find

Find a substring inside a text object.

Prototype:

int FPDFTextObj_Find(FPDF_PAGEOBJECT text_object, unsigned short* text, int nwSize, int nStart = 0);

Parameters:

text_object - Handle of text object returned by FPDFPageObj_NewTextObj or FPDFPageObj_NewTextObjEx.

text - A pointer to a string to search for.

nwSize - The length of the string.

nStart - The index of the character in the string to begin the search with,

or 0 to start from the beginning.

Return Value:

Return the zero-based index of the first character in this text object that matches the requested substring or characters; -1 if the substring or character is not found.

FPDFTextObj_Replace

Replace a character with another.

Prototype:

int FPDFTextObj_Replace(FPDF_PAGEOBJECT text_object, unsigned short* OldText, int nwOldSize, unsigned short* NewText, int nwNewSize);

Parameters:

text_object	-	Handle of text object returned by FPDFPageObj_NewTextObj or FPDFPageObj_NewTextObjEx.
OldText	-	A pointer to a string containing the character to be replaced by NewText.
nwOldSize	-	The length of the old text.
NewText	-	A pointer to a string containing the character replacing OldText.
nwNewSize	-	The length of the new text.

Return Value:

Return the number characters of the changed text object. -1 if the text object isn't changed.

FPDFTextObj_Compare

Compare current text object characters with another string.

Prototype:

FPDF_BOOL FPDFTextObj_Compare(FPDF_PAGEOBJECT text_object, unsigned short* text, int nwSize);

Parameters:

text_object	-	Handle of text object returned by FPDFPageObj_NewTextObj or FPDFPageObj_NewTextObjEx
text	-	The string used for comparison.
nwSize	-	The length of the string.

Return Value:

TRUE if the texts are identical, and FALSE if they are different.

e) Fonts Operation

FPDFFont_GetFontName

Retrieve the face name of the font and return the name in a string.

Prototype:

FPDF_BYTESTRING FPDFFont_GetFontName(FPDF_FONT font);

Parameters:

font - Handle of the font object.

Return Value:

Pointer to the string.

FPDFFont_GetFlags

Get the value of the flags entry in a font descriptor.

Prototype:

int FPDFFont_GetFlags(FPDF_FONT font);

Parameters:

font - Handle of the font object.

Return Value:

The value of the Flags entry defined following.

```
// All glyphs have the same width.
FPDF_FLAG_FIXEDPITCH        1
// Glyphs have serifs, which are short strokes drawn at an angle at the top and bottom of //
glyph stems.
FPDF_FLAG_SERIF             2
// Font contains glyphs outside the Adobe standard Latin character set. This flag and the
// Nonsymbolic flag cannot both be set or both be clear.
FPDF_FLAG_SYMBOLIC          4
// Glyphs resemble cursive handwriting.
FPDF_FLAG_SCRIPT            8
// Font uses the Adobe standard Latin character set or a subset of it.
FPDF_FLAG_NONSYMBOLIC       32
// Glyphs have dominant vertical strokes that are slanted.
FPDF_FLAG_ITALIC            64
// Font contains no lowercase letters; typically used for display purposes, such as for titles
// or headlines.
FPDF_FLAG_ALLCAP            0x10000
// Font contains both uppercase and lowercase letters.
FPDF_FLAG_SMALLCAP          0x20000
// Whether bold glyphs are painted with extra pixels even at very small text sizes.
FPDF_FLAG_FORCEBOLD         0x40000
```

Comment:

The value of the Flags entry in a font descriptor is an unsigned 32-bit integer containing flags specifying various characteristics of the font.

FPDFFont_GetBBox

Get the font bounding box.

Prototype:

Void FPDFFont_GetBBox(FPDF_FONT font, double* left, double* right, double* bottom, double* top);

Parameters:

font - Handle of the font object.
left - Pointer to a double value receiving the left of a rectangle (in points).
right - Pointer to a double value receiving the right of a rectangle (in points).
bottom - Pointer to a double value receiving the bottom of a rectangle (in points).
top - Pointer to a double value receiving the top of a rectangle (in points).

Return Value:

None.

FPDFFont_GetCharBBox

Get the special character bounding box of a font object.

Prototype:

Void FPDFFont_GetCharBBox(FPDF_FONT font, FPDF_DWORD char_code, double* left, double* right, double* bottom, double* top);

Parameters:

font - Handle of the font object.
char_code - The character code.
left - Pointer to a double value receiving the left of a rectangle (in points).
right - Pointer to a double value receiving the right of a rectangle (in points).
bottom - Pointer to a double value receiving the bottom of a rectangle (in points).
top - Pointer to a double value receiving the top of a rectangle (in points).

Return Value:

None.

FPDFFont_GetCharWidth

Get a special character width of a font object.

Prototype:

int FPDFFont_GetCharWidth(FPDF_FONT font, FPDF_DWORD char_code);

Parameters:

font - Handle of the font object.
char_code - The character code.

Return Value:

The value of the width.

FPDFFont_GetUnicode

Get the unicode value from a character code.

Prototype:

unsigned short FPDFFont_GetUnicode(FPDF_FONT font, FPDF_DWORD char_code);

Parameters:

font - Handle of the font object.

char_code - The character code.

Return Value:

The unicode value.

FPDFFont_GetCharSize

Get the character size in the font.

Prototype:

int FPDFFont_GetCharSize(FPDF_FONT font, FPDF_DWORD char_code);

Parameters:

font - Handle of the font object.

char_code - The character code.

Return Value:

The value of the character size.

FPDFFont_GetCharCode

Get the character code from a unicode value.

Prototype:

FPDF_DWORD FPDFFont_GetCharCode(FPDF_FONT font, unsigned short unicode);

Parameters:

font - Handle of the font object.

unicode - The unicode of a special character.

Return Value:

The character code value.

FPDFFont_GetFontDescent

Get the descent value in the font.

Prototype:

int FPDFFont_GetFontDescent (FPDF_FONT font);

Parameters:

font - Handle of the font object.

Return Value:

The descent value.

Comment:

The descent value describes the maximum depth below the baseline reached by glyphs in this font. The value is a negative number.

FPDFFont_GetFontAscent

Get the ascent value in the font.

Prototype:

int FPDFFont_GetFontAscent(FPDF_FONT font);

Parameters:

font - Handle of the font object.

Return Value:

The ascent value.

Comment:

The ascent value describes the maximum height above the baseline reached by glyphs in this font, excluding the height of glyphs for accented characters.

FPDFFont_AddTrueType

Create a true type font and add the font into the document.

Prototype:

FPDF_FONT FPDFFont_AddTrueType(FPDF_DOCUMENT document, PFPDFLOGFONT pLogFont);

Parameters:

document - Handle to document. Returned by FPDF_LoadDocument and FPDF_CreateNewDocument.

pLogFont - The tagFPDFLOGFONT structure defined following.

```
typedef struct tagFPDFLOGFONT {
    long lfWeight;
    // Specifies the height, in logical units, of the character cell or character of the font.
    unsigned char lfItalic;
    // Specifies the average width, in logical units, of characters in the font.
    unsigned char lfCharSet;
    // Specifies the character set.
    unsigned char lfPitchAndFamily;
    // Specifies the pitch and family of the font.
    char lfFaceName[32];
    // A null-terminated string that specifies the typeface name of the font.
} FPDFLOGFONT, *PFPDFLOGFONT;
```

Return Value:

Handle of the new font object.

FPDFFont_AddStandardFont

Create a standard type font and add the font into the document.

Prototype:

FPDF_FONT FPDFFont_AddStandardFont(FPDF_DOCUMENT document, char* font_name, int encoding);

Parameters:

document - Handle to document. Returned by FPDF_LoadDocument and FPDF_CreateNewDocument.

font_name - A null-terminated string that specifies the typeface name of the font.

encoding - The font encoding defined following.

```
// Built-in encoding.
FPDF_ENCODING_DEFAULT       0
// WinAnsiEncoding.
FPDF_ENCODING_WINANSI       1
// MacRomanEncoding.
FPDF_ENCODING_MACROMAN      2
// MacExpertEncoding.
FPDF_ENCODING_STANDARD      3
// StandardEncoding. Adobe standard encoding.
FPDF_ENCODING_MACEXPERT     4
// Not support yet. Reserved.
FPDF_ENCODING_IDENTITY      10
// Not support yet. Reserved.
FPDF_ENCODING_GB            11
// Not support yet. Reserved.
FPDF_ENCODING_BIG5          12
// Not support yet. Reserved.
FPDF_ENCODING_SHIFT_JIS     13
// Not support yet. Reserved.
FPDF_ENCODING_KOREAN        14
```

Return Value:

Handle of the new font object.

f) Path Operation

FPDFPageObj_NewPathObj

Create a new path Object.

Prototype:

FPDF_PAGEOBJECT FPDFPageObj_NewPathObj(void);

Parameters:

None.

Return Value:

Handle of path object.

FPDFPathObj_SetPoints

Set the points in a path object.

Prototype:

void FPDFPathObj_SetPoints(FPDF_PAGEOBJECT path_object, int point_count, double* point_x, double* point_y, char* flags);

Parameters:

path_object - Handle of path object returned by FPDFPageObj_NewPathObj.
point_count - The total number of points
point_x - The pointer to point at an array of X coordinate of points
point_y - The pointer to point at an array of Y coordinate of points.
flags - The pointer to point at an array of the path point types defined following.

FPDF_PATH_CLOSEFIGURE	1
FPDF_PATH_LINETO	2
FPDF_PATH_BEZIERTO	4
FPDF_PATH_MOVETO	6

Return Value:

None.

FPDFPathObj_InsertPoint

Insert a point to a path object.

Prototype:

void FPDFPathObj_InsertPoint(FPDF_PAGEOBJECT path_object, double x, double y, int flag, int point_index = -1);

Parameters:

path_object - Handle of path object returned by FPDFPageObj_NewPathObj.
x - The X coordinate of a point.
y - The Y coordinate of a point.
flag - The path point types defined following.

FPDF_PATH_CLOSEFIGURE	1
FPDF_PATH_LINETO	2
FPDF_PATH_BEZIERTO	4
FPDF_PATH_MOVETO	6

point_index - The index of the point.

Return Value:

None.

FPDFPathObj_RemovePoint

Remove a point in a path object.

Prototype:

void FPDFPathObj_RemovePoint(FPDF_PAGEOBJECT path_object, int point_index);

Parameters:

path_object - Handle of path object returned by FPDFPageObj_NewPathObj.

point_index - The index of the point

Return Value:

None.

FPDFPathObj_ClearPoints

Clear the points in a path object.

Prototype:

void FPDFPathObj_ClearPoints(FPDF_PAGEOBJECT path_object);

Parameters:

path_object - Handle of path object returned by FPDFPageObj_NewPathObj.

Return Value:

None.

FPDFPathObj_CountPoints

Get the number of points in a path object.

Prototype:

int FPDFPathObj_CountPoints(FPDF_PAGEOBJECT path_object);

Parameters:

path_object - Handle of path object returned by FPDFPageObj_NewPathObj.

Return Value:

The point count in a path object.

FPDFPathObj_GetPointType

Get the type of a special point.

Prototype:

int FPDFPathObj_GetPointType(FPDF_PAGEOBJECT path_object, int point_index);

Parameters:

path_object - Handle of path object returned by FPDFPageObj_NewPathObj.

point_index - The index of the point.

Return Value:

The path point types defined following.

FPDF_PATH_CLOSEFIGURE	1
FPDF_PATH_LINETO	2
FPDF_PATH_BEZIERTO	4
FPDF_PATH_MOVETO	6

FPDFPathObj_GetPointX

Get the X coordinate of a specific point.

Prototype:

double FPDFPathObj_GetPointX(FPDF_PAGEOBJECT path_object, int point_index);

Parameters:

path_object - Handle of path object returned by FPDFPageObj_NewPathObj.

point_index - The index of the inserted point

Return Value:

The value of the X coordinate of the point.

FPDFPathObj_GetPointY

Get the Y coordinate of a special point.

Prototype:

double FPDFPathObj_GetPointY(FPDF_PAGEOBJECT path_object, int point_index);

Parameters:

path_object - Handle of path object returned by FPDFPageObj_NewPathObj.

point_index - The index of the point

Return Value:

The value of the Y coordinate of the point.

FPDFPathObj_GetPath

Get the path in a path object.

Prototype:

FPDF_PATH FPDFPathObj_GetPath(FPDF_PAGEOBJECT path_object);

Parameters:

path_object - Handle of path object returned by FPDFPageObj_NewPathObj.

Return Value:

The path in the path object.

FPDFPathObj_GetStroke

Get the stroke mode of the path.

Prototype:

FPDF_BOOL FPDFPathObj_GetStroke(FPDF_PAGEOBJECT path_object);

Parameters:

path_object - Handle of path object returned by FPDFPageObj_NewPathObj.

Return Value:

Return the stroke mode. TRUE for stroking the line, FALSE for no stroking.

FPDFPathObj_GetFillMode

Get the fill method of the path.

Prototype:

int FPDFPathObj_GetFillMode(FPDF_PAGEOBJECT path_object);

Parameters:

path_object - Handle of path object returned by FPDFPageObj_NewPathObj.

Return Value:

Returns fill mode types defined following.

FPDF_FILL_NULL	0
FPDF_FILL_ALTERNATE	1
FPDF_FILL_WINDING	2

FPDFPathObj_SetFillStroke

Set the fill and stroke mode of the path.

Prototype:

void FPDFPathObj_SetFillStroke(FPDF_PAGEOBJECT path_object, int fill_type, int stroke);

Parameters:

path_object - Handle of path object returned by FPDFPageObj_NewImgeObj.

fill_type - The fill mode enumeration defined following.

FPDF_FILL_NULL	0
FPDF_FILL_ALTERNATE	1
FPDF_FILL_WINDING	2

stroke - The stroke mode. TRUE for stroking the line, FALSE for no stroking.

Return Value:

None.

FPDFPathObj_GetMatrix

Get the matrix of the path.

Prototype:

FPDF_BOOL FPDFPathObj_GetMatrix(FPDF_PAGEOBJECT path_object,
double* a, double* b, double* c, double* d,double* e, double* f);

Parameters:

path_object	-	Handle of path object returned by FPDFPageObj_NewPathObj.
a	-	Pointer to a double value receiving the coefficient "a" of the matrix.
b	-	Pointer to a double value receiving the coefficient "b" of the matrix.
c	-	Pointer to a double value receiving the coefficient "c" of the matrix.
d	-	Pointer to a double value receiving the coefficient "d" of the matrix.
e	-	Pointer to a double value receiving the coefficient "e" of the matrix.
f	-	Pointer to a double value receiving the coefficient "f" of the matrix.

Return Value:

TRUE for successful, FALSE for failed.

FPDFPathObj_SetMatrix

Set the matrix of the path.

Prototype:

FPDF_BOOL FPDFPathObj_SetMatrix(FPDF_PAGEOBJECT path_object,
double a, double b, double c, double d, double e, double f);

Parameters:

image_object	-	Handle of path object returned by FPDFPageObj_NewPathObj.
a	-	The coefficient "a" of the matrix.
b	-	The coefficient "b" of the matrix.
c	-	The coefficient "c" of the matrix.
d	-	The coefficient "d" of the matrix.
e	-	The coefficient "e" of the matrix.
f	-	The coefficient "f" of the matrix.

Return value:

TRUE for successful, FALSE for failed.

g) Forms Operation

FPDFFormObj_InsertSubObject

Insert a page object to the form object. The page object is automatically freed.

Prototype:

Void FPDFFormObj_InsertSubObject(FPDF_PAGEOBJECT form_object,
FPDF_PAGEOBJECT page_obj);

Parameters:

form_object	-	Handle to a form object. Returned by FPDFPage_GetObject.
page_obj	-	Handle to a page object. Returned by FPDFPageObj_NewTextObj, FPDFPageObj_NewTextObjEx, FPDFPageObj_NewPathObj, FPDFPageObj_ImageObj, etc.

Return value:

None.

FPDFFormObj_Display

Render contents in a form_object to a device (screen, bitmap, or printer).

Prototype:

void FPDFFormObj_Display(FPDF_PAGEOBJECT form_object,void* device,
int xPos,int yPos,int xSize, int ySize,int iRotate);

Parameters:

device - Handle to device context.
form_object - Handle to the form_object. Returned by FPDFPage_GetObject function.
xPos - Left pixel position of the display area in the device coordinate.
yPos - Top pixel position of the display area in the device coordinate.
xSize - Horizontal size (in pixels) for displaying the page.
ySize - Vertical size (in pixels) for displaying the page.
iRotate - Page orientation: 0 (normal), 1 (rotated 90 degrees clockwise), 2 (rotated 180 degrees), 3 (rotated 90 degrees counter-clockwise).

Return value:

None.

FPDFFormObj_DeleteSubObject

Delete a page object from the form object. The page object is automatically freed.

Prototype :

Void FPDFFormObj_DeleteSubObject(FPDF_PAGEOBJECT form_object, int index);

Parameters:

form_object - Handle to a form object. Returned by FPDFPage_GetObject.
index - The index of a page object.

Return value:

None.

FPDFFormObj_CountSubObjects

Get number of page objects inside the form object.

Prototype:

int FPDFFormObj_CountSubObjects(FPDF_PAGEOBJECT form_object);

Parameters:

form_object - Handle to a form object. Returned by FPDFPage_GetObject.

Return value:

The number of the page objects.

FPDFFormObj_GetSubObject

Get the page object from a form object.

Prototype:

FPDF_PAGEOBJECT FPDFFormObj_GetSubObject(FPDF_PAGEOBJECT form_object, int index);

Parameters:

form_object - Handle to a form object. Returned by FPDFPage_GetObject.
index - The index of a page object.

Return value:

The handle of the page object. Null for failed.

FPDFFormObj_GetSubObjectIndex

Get the index of the page object from the specified form object.

Prototype:

int FPDFFormObj_GetSubObjectIndex(FPDF_PAGEOBJECT form_object,
FPDF_PAGEOBJECT page_obj);

Parameters:

form_object - Handle to a form object. Returned by FPDFPage_GetObject.

page_obj - Handle to a page object. Returned by FPDFPageObj_NewTextObj, FPDFPageObj_NewTextObjEx,FPDFPageObj_NewPathObj, etc.

Return value:

If successful, returns the index of the page object.

If failed, it return -1.

FPDFFormObj_GenerateContent

Generate content of the form object.

Prototype:

FPDF_BOOL DFFormObj_GenerateContent(FPDF_PAGEOBJECT form_object)

Parameters:

form_object - Handle to a form object. Returned by FPDFPage_GetObject.

Return value:

True for successful, and false for failed.

Comment:

When the form object's information is changed, you must call the FPDFFormObj_GenerateContent function. Or the changed information will be lost.

FPDFFormObj_GetMatrix

Get the form object matrix.

Prototype:

FPDF_BOOL FPDFFormObj_GetMatrix(FPDF_PAGEOBJECT form_object,
double* a, double* b, double* c, double* d,double* e, double* f)

Parameters:

form_object - Handle to a form object. Returned by FPDFPage_GetObject.

a - Pointer to a double value receiving the coefficient "a" of the matrix.

b - Pointer to a double value receiving the coefficient "b" of the matrix.

c - Pointer to a double value receiving the coefficient "c" of the matrix.

d - Pointer to a double value receiving the coefficient "d" of the matrix.

e - Pointer to a double value receiving the coefficient "e" of the matrix.

f - Pointer to a double value receiving the coefficient "f" of the matrix.

Return Value:

TRUE for successful, FALSE for failed.

h) Images Operation

FPDFPageObj_NewImgeObj

Create a new Image Object.

Prototype:

FPDF_PAGEOBJECT FPDFPageObj_NewImgeObj(FPDF_DOCUMENT document);

Parameters:

document - Handle to document. Returned by FPDF_LoadDocument or FPDF_CreateNewDocument .

Return Value:

Handle of image object.

FPDFImageObj_LoadFromFile

Load Image from a image file(such as *.bmp,*.jpeg) and then set it to an image object.

Prototype:

FPDF_BOOL FPDFImageObj_LoadFromFile(FPDF_PAGE* pages, int nCount, FPDF_PAGEOBJECT image_object, FPDF_BYTESTRING filename);

Parameters:

pages - Pointers to the start of all loaded pages.

nCount - Number of pages.

image_object - Handle of image object returned by FPDFPageObj_NewImgeObj.

filename - The full path of the image file.

Return Value:

TRUE if successful, FALSE otherwise.

FPDFImageObj_GetMatrix

Get the matrix of an image object.

Prototype:

FPDF_BOOL FPDFImageObj_GetMatrix(FPDF_PAGEOBJECT image_object, double* a, double* b, double* c, double* d,double* e, double* f);

Parameters:

image_object - Handle of image object returned by FPDFPageObj_NewImgeObj.

a - Pointer to a double value receiving the coefficient "a" of the matrix.

b - Pointer to a double value receiving the coefficient "b" of the matrix

c - Pointer to a double value receiving the coefficient "c" of the matrix

d - Pointer to a double value receiving the coefficient "d" of the matrix

e - Pointer to a double value receiving the coefficient "e" of the matrix

f - Pointer to a double value receiving the coefficient "f" of the matrix

Return value:

TRUE if successful, FALSE otherwise.

Comments:

A matrix defines transformation of coordinate from one space to another.

In PDF, a matrix is defined by the following equations:

x' = a * x + c * y + e;

y' = b * x + d * y + f;

FPDFImageObj_GetMatrix function is used to get a,b,c,d,e,f coefficients of the transformation from "image space" to "user space".

FPDFImageObj_SetMatrix

Set the matrix of an image object.

Prototype:

FPDF_BOOL FPDFImageObj_SetMatrix(FPDF_PAGEOBJECT image_object,
double a, double b, double c, double d, double e, double f);

Parameters:

image_object - Handle of image object returned by FPDFPageObj_NewImgeObj.

a - The coefficient "a" of the matrix.

b - The coefficient "b" of the matrix.

c - The coefficient "c" of the matrix.

d - The coefficient "d" of the matrix.

e - The coefficient "e" of the matrix.

f - The coefficient "f" of the matrix.

Return value:

TRUE if successful, FALSE otherwise.

FPDFImageObj_GetBitmap

Get the bitmap from an image object.

Prototype:

FPDF_BITMAP FPDFImageObj_GetBitmap(FPDF_PAGEOBJECT image_object)

Parameters:

image_object - Handle of image object returned by FPDFPageObj_NewImgeObj.

Return value:

The handle of the bitmap.

FPDFImageObj_SetBitmap

Set the bitmap to an image object.

Prototype:

FPDF_BOOL FPDFImageObj_SetBitmap(FPDF_PAGE* pages,int nCount,
FPDF_PAGEOBJECT image_object, FPDF_BITMAP bitmap);

Parameters:

pages - Pointer's to the start of all loaded pages.

nCount - Number of pages.

image_object - Handle of image object returned by FPDFPageObj_NewImgeObj.

bitmap - The handle of the bitmap which you want to set it to the image object.

Return value:

TRUE if successful, FALSE otherwise.

Redistribution

NOTE: You can only redistribute files listed below to customers of your application, under a written SDK license agreement with Foxit. You cannot distribute any part of the SDK to general public, even with a SDK license agreement. Without SDK license agreement, you cannot redistribute anything.

For better performance and rendering quality, some Windows system DLL may also be redistributed with FOXIT PDF SDK, redistribution of such Windows DLLs is subject to redistribution license agreement with Microsoft. You can visit Microsoft website for more information.

Operating System	Foxit DLL	System DLL
Windows XP or later	FPDFSDK.DLL	None
Windows 98/Me/2000	FPDFSDK.DLL	GDIPLUS.DLL
Windows 95/NT	FPDFSDK.DLL	IE 4.0 or later required
Linux	libfpdfsdk.so	None

To get Windows DLLs, please visit Microsoft website or get it from a system running other OS. If you have any questions, please feel free to contact us at support@foxitsoftware.com.