DEMO TUTORIAL

Foxit® PDF SDK

Microsoft® Partner
Gold Independent Software Vendor (ISV)

Table of Contents

1 ABOUT FOXIT PDF SDK

Foxit PDF SDK is a powerful, secure and efficient PDF software development kit for application developers to view, search, and annotate PDF documents and forms by using Foxit PDF technology. Foxit PDF SDK is easy to integrate and platform independent. It reduces time for releasing by developing features and porting them to multiple platforms.

1.1 Functions provided in Foxit PDF SDK

Foxit PDF SDK provides the following functions:

1) **Basic operation functions, defined in header file fs_base_r.h:**
    a) Library management: includes Foxit PDF SDK libraries related functions like initialization, license applying, memory management, logs etc.
    b) Character strings manipulation: includes input/output functions. Most functions use UTF-8 as character encoding format. Character strings can also be represented by byte strings.
    c) File access: provides a file access system which is based on file streams for the needs of various data resources.
    d) Multiple threads support: used for operations under multiple-thread environment.
    e) Common data structure: includes data structures such as dates, rectangles, matrices, paths etc., and definitions of related manipulator functions.
    f) Progressive control: provides related functions about progressive PDF processing which usually needs to take more time to finish. To help applications supporting progressive processing, Foxit PDF SDK has designed effective and flexible APIs for applications to call.
    g) Font manipulation: includes font loading, font data accessing and font substitution.

2) **Codecs, defined in header file fs_codec_r.h:**

    Currently supported codecs include UTF-8, Base-64, digest algorithm and Flate. Foxit PDF SDK provides related APIs for applications to access them.

3) **Image support, defined in two header files fs_image_r.h and fs_image_w.h:**

    Foxit PDF SDK provides APIs for bitmap processing, extraction and generation.

4) **Application extensions, defined in header file fs_app_r.h:**

    Foxit PDF SDK provides this for applications to extend functionalities. In the coming releases, more functions in Foxit PDF SDK need to work together with applications to meet customers' requirements. Usually, it requires Foxit PDF SDK to report its status to applications through some events. For example, PSI in Foxit PDF SDK informs applications to do refresh through an event.

5) **Barcode, defined in header file fs_barcode_w.h:**

   Barcode is an independent module from outside and it needs applications to perform barcode module initialization. Currently, Foxit PDF SDK supports 8 types of barcode formats.

6) **PDF manipulation, defined in the header files fpdf_base_r.h, fpdf_document_w.h, fpdf_page_r.h, fpdf_page_w.h etc. and initialization module that defined in header file fpdf_base_r.h is required. The functions include:**

   a) Basic PDF reading. Related header files are fpdf_base_r.h, fpdf_document_r.h, fpdf_document_w.h, fpdf_page_r.h, fpdf_page_w.h etc. It provides basic operations like loading PDF file, basic document data access, loading page, rendering page content, saving PDF file, metadata manipulation, listing directory, reading reader properties, attachment access etc.

   b) PDF asynchronous loading, defined in header file fpdf_async_r.h. It is used to load remote files on internet through asynchronous access mode.

   c) PDF annotation manipulation, defined in header files fpdf_annot_r.h and fpdf_annot_w.h. It includes operations like accessing properties, loading, creating and deleting annotations etc.

   d) PDF form manipulation, defined in header files fpdf_form_r.h, fpdf_form_w.h, ffdf_document_r.h, and ffdf_document_w.h. It includes operations like loading forms, retrieving/setting form data, displaying form control, creating or removing form fields, and setting or reading the values and properties of the form fields etc. Currently, the forms manipulation focuses on data processing and form design. It doesn’t provide interfaces to run script, but can execute the script embedded in a document.

   e) PDF page content edition, defined in the header files fpdf_pageobjects_r.h, and fpdf_pageobjects_w.h. It includes operations like retrieving, creating and modifying page content. PDF pages have four primitive types: text, path, image and form XObject.

   f) PDF security, defined in header files fpdf_security_r.h, and fpdf_security_w.h. It provides APIs for encryption of PDF files.

   g) Microsoft RMS encryption and decryption, defined in header files fpdf_security_r.h, and fpdf_security_w.h. It provides APIs to integrate with Microsoft RMS encryption and decrypiton which allows developers to work with the Microsoft RMS SDK to both encrypt and decrypt PDF documents.

   h) PDF reflowing, defined in header file fpdf_reflow_r.h. It provides APIs for reflowing PDF pages. This feature is specially designed for the needs of mobile devices and small screen display.

i) PDF text manipulation, defined in header file fpdf_textpage_r.h. It includes operations like extracting, selecting, searching PDF text and recognizing hyperlinks.

j) PDF watermark manipulation, defined in header file fpdf_watermark_w.h. It includes operations like creating watermarks from text, bitmap, image, or PDF page.

k) PDF objects manipulation, defined in header files fpdf_objects_r.h and fpdf_objects_w.h. It provides general PDF object access methods and could be used to meet advanced requirements.

l) PDF layer, defined in header file fpdf_layer_r.h. It provides APIs to allow users to render all PDF layers or view or hide the contents in each layer selectively.

7) **Pressure handwriting, defined in header file fs_psi_w.h:**

Foxit PDF SDK provides effects of special handwriting pressure processing and handwriting simulation display. Usually, it can be used for handwriting signature.

8) **Universal displaying, defined in header files fs_renderer_r.h, fs_renderer_windows_r.h (only for Windows), and fs_renderer_apple_r.h (only for Mac and iOS):**

Foxit PDF SDK provides universal drawing engines for display. All display APIs in Foxit PDF SDK are based on this functionality.

1.2 Steps for using Foxit PDF SDK

The following steps should be done when using Foxit PDF SDK:

1) Initialize library. Call FSCRT_Library_CreateMgr or FSCRT_Library_CreateDefaultMgr to create memory managers. The latter one is suggested for a desktop environment, a server, or other large memory devices.

2) Apply a License. You should call FSCRT_License_UnlockLibrary to unlock the library. Foxit PDF SDK APIs cannot be called normally until license is checked and valid.

3) Initialize modules. Call FSCRT_PDFModule_Initialize to initialize PDF module for using PDF-related functions. If to barcode-related functions, FSCRT_BCModule_Initialize should be called to initialize barcode module.

4) Release resources. Modules should be finalized first, by calling FSCRT_PDFModule_Finalize for PDF module, or FSCRT_BCModule_Finalize for barcode module. Then call FSCRT_Library_DestroyMgr to destroy memory managers.

2 ABOUT DEMOS

In order to help users to develop applications based on Foxit PDF SDK, Foxit PDF SDK provides some demos to show how to implement some important PDF functionalities by calling Foxit PDF SDK APIs. These simple demos support the platforms of Windows, Linux, Mac and iOS.

2.1 Getting started

Currently, demos are implemented as console applications without user interfaces. These demos are developed based on C/C++ and some necessary comments are provided to help developers understand the logics in demo's source code.

2.2 Directory structure for SDK package and demos

1) docs: contains documents with introduction of demos' feature, how to compile and run demos and API references of Foxit PDF SDK etc.
2) include: stores header files of Foxit PDF SDK.
3) lib (or"libs"): stores library files and key files of Foxit PDF SDK.
4) samples/simple_sample
    a) Directory bin: stores executable files. They are organized in project configuration-named subdirectories.
    b) Directory input_files: stores files which are used as default input files for running demos.
    c) Directory output_files: stores output files of the demos along with log files.
    d) Directory comm_src: stores public source files which are used in all demos.
    e) all_samples_vc2008.sln: all demo projects targeting vc2008. This file is used to compile all demos under VS2008 in Windows.
    f) all_samples_vc2010.sln: all demo projects targeting vc2010. This file is used to compile all demos under VS2010 in Windows.
    g) fxconfig: a configuration of make-file which is used to compile all demos in Linux and Mac.
    h) Makefile: a collection of make-file which is used to compile all demos in Linux and Mac.
    i) Directory simple_sample_ios: all demo projects targeting Xcode 5.0. The "simple_sample_ios.xcodeproj" file is used to build all demos under Xcode in iOS.

2.3 Compiling demos

There are 4 kinds of projects available for each demo:

1) **Project for Microsoft Visual Studio 2008**: It can be used to compile demos in both 32-bit and 64-bit system. In 32-bit system, generated executable files with release and debug version will be respectively stored in “bin/rel_x86_vc8” and “bin/dbg_x86_vc8”. In 64-bit system, generated executable files will be respectively stored in “bin/rel_x64_vc8” and “bin/dbg_x64_vc8”.

2) **Project for Microsoft Visual Studio 2010**: It can be used to compile demos in both 32-bit and 64-bit. In 32-bit system, generated executable files with release and debug version will be respectively stored in “bin/rel_x86_vc2010” and “bin/dbg_x86_vc2010”. In 64-bit system, generated executable files will be respectively stored in “bin/rel_x64_vc2010” and “bin/dbg_x64_vc2010”.

3) **Make file for GCC**. It is used in Linux/MAC environment to compile demos with GCC compilers. Developers should choose a proper configuration based on their developing environment. By default, generated executable files with release and debug version will be respectively stored in “bin/rel_gcc” and “bin/dbg_gcc”.

File “makefile” in directory “simple_sample” is used for compiling all demos. Use command “make” to compile, while use command “make clean” to clean. Alternatively, execute command “make” directly in directory for each sample to compile corresponding sample.

File “fxconfig” in directory “simple_sample” is stored configuration for building. User can modify it based on their own settings. The specific instructions of this file are shown below:

```
ver=release
SRC=comm_src
CXX=g++

INCLUDE_PATH=-L../../include
FXLIB_PATH=../../lib

DEST_PATH=./bin/dbg_gcc
OBJ_PATH=./obj/dbg
CCFLAGS=-g –c
```

ver=release: a release version.

SRC: path to the directory of common source files.

INCLUDE_PATH: path to the directory storing include files of Foxit PDF SDK.

CXX: path specifying the directory of compliers.

FXLIB_PATH: path specifying the directory where Foxit PDF SDK libraries are.

DEST_PATH: path of output. Make sure that the directory specified by DEST exists.

OBJ_PATH: path of output obj files.

CCFLAGS: set the compile options.

4) **Project for Xcode 5.0**: It is used in iOS environment to build demos with Xcode. Developers should open the ".xcodeproj" file to load the all demo projects and build them. Generated executable files will be stored in the "Sandbox".

Before compiling demos, users should make sure the header files and library files of Foxit PDF SDK are available in directory "include" and "lib" respectively.

2.4 Running the demos

Before running demos, please ensure that the test files are placed in default input directory which are not protected by password encryption or other security solutions. Each demo will output logs directly to command prompt window or terminal and also save to a log file named "log.txt" in output directory except matrix demo.

For Windows, demos can be run directly in Microsoft Visual Studio environment or in command prompt window with command. **For Linux/MAC**, demos should be run in command terminal. **For iOS**, after building the demos in Xcode, if you choose to run in a simulator, the simulator will start.

2.5 Common functions among demos

In directory "comm_src", there are two files named fgsdk_common.h and fgsdk_common.cpp. These files provide declarations and implementations of common functions used by all demos. These common functions are implemented based on Foxit PDF SDK and STL and they can be compiled in different platforms. Table 2-1 lists the common SDK functions (APIs) that are used in all demos, and other functions will be introduced in each demo. Note the function FSCRT_BStr_Set is not used in matrix demo.

Table 2-1

API name	Description
FSCRT_Library_CreateDefaultMgr	Initialize Foxit PDF SDK Library and create a default manager
FSCRT_Library_CreateMgr	Initialize Foxit PDF SDK Library and create an extension manager
FSCRT_Library_DestroyMgr	Release Foxit PDF SDK Library and destroy the current manager

FSCRT_BStr_Init	Initialize character string objects
FSCRT_BStr_Set	Set character string data
FSCRT_BStr_Clear	Clear character string objects
FSCRT_File_CreateFromFileName	Create a file object from a given file name
FSCRT_File_Create	Create a file object (by file processor)
FSCRT_File_Release	Release a file object
FSCRT_Progress_Continue	Continuously calling a step-by-step progress
FSCRT_Progress_Release	Release a step-by-step progress

3 DEMOS DETAILS

This section respectively introduces the demos in detail.

3.1 Pdf2text

3.1.1 Description

1) Purpose: illustrates how to use Foxit PDF SDK to extract the text from PDFfiles and save them as text files.
2) Property: this demo is a console application without UI.

   It supports command line argument.

   Usage: *pdf2text [<srcfile> /r [pagerange] /o [destfolder]]*

<srcfile>	Path of a PDF file as input file
/r [pagerange]	Page range, used to specify indexes of PDF pages in input PDF file, such as "0", "1-3, 5". Page index starts from 0 and should be valid within page count. Invalid index will be ignored. If input page range is totally invalid, page range will be set to first page by default.
/o [destfolder]	Path of a folder as output directory. If not set, use default output directory.

3) Process: the demo loads specified PDF files from input directory. For each PDF file, this demo will extract the texts of it and then save as text file. Once completed, the demo will exit. All the information about execution process will be recorded into log file.
4) Input directory: used to place input files.
   a) Default: /samples/simple_sample/input_files. Here're two ways:
      i. Use default files: AboutFoxit.pdf.
      ii. Set default sub-folder: create a sub-folder named "pdf2text" and put all input files into this sub-folder. When running, this demo will use files in sub-folder as input files.
   b) User-defined: set single input file through command line argument.
5) Output directory: used to store output files, including log.txt.
   a) Default: /samples/simple_sample/output_files/pdf2text.
   b) User-defined: set path of output folder through command line argument and output

directory will be “/outputfolder/pdf2text”.

6) Program files:

pdf2text.cpp:	source code for retrieving text from PDF and saving as text file.

3.1.2 SDK functions

The demo of pdf2text uses following additional functions of Foxit PDF SDK:

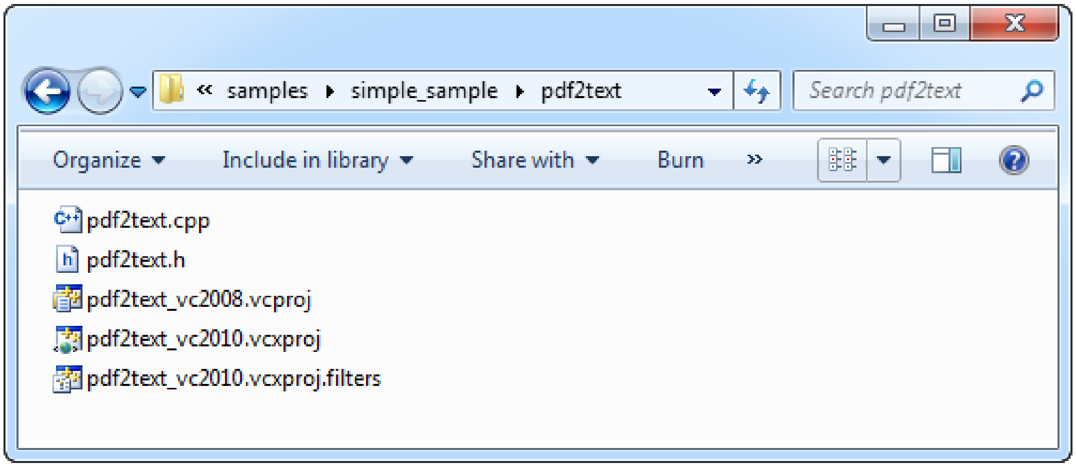
1) FSPDF_Doc_LoadFromFile: used for loading PDF documents.
2) FSPDF_Doc_CountPages: used for counting the pages of PDF documents.
3) FSPDF_Doc_GetPage: used for getting a PDF page object.
4) FSPDF_Page_Clear: used for clearing a PDF page object and releasing page resources without destroying the page object.
5) FSPDF_Doc_Close: used for closing PDF documents.
6) FSPDF_Page_StartParse: used for starting to parse a PDF page object and returning a progress object.
7) FSPDF_TextPage_Load: used for loading text page objects.
8) FSPDF_TextPage_ExportToFile: used for saving text page objects as text files.
9) FSPDF_TextPage_Release: used for releasing text page objects.
10) FSCRT_PDFModule_Initialize: used for initializing PDF modules.
11) FSCRT_PDFModule_Finalize: used for releasing PDF modules.

3.1.3 Run

In this tutorial, one thing to note is that the highlighted rectangles in the figures are the version of the SDK. Here the SDK version is 5.3, so it shows 5_3.

For Windows

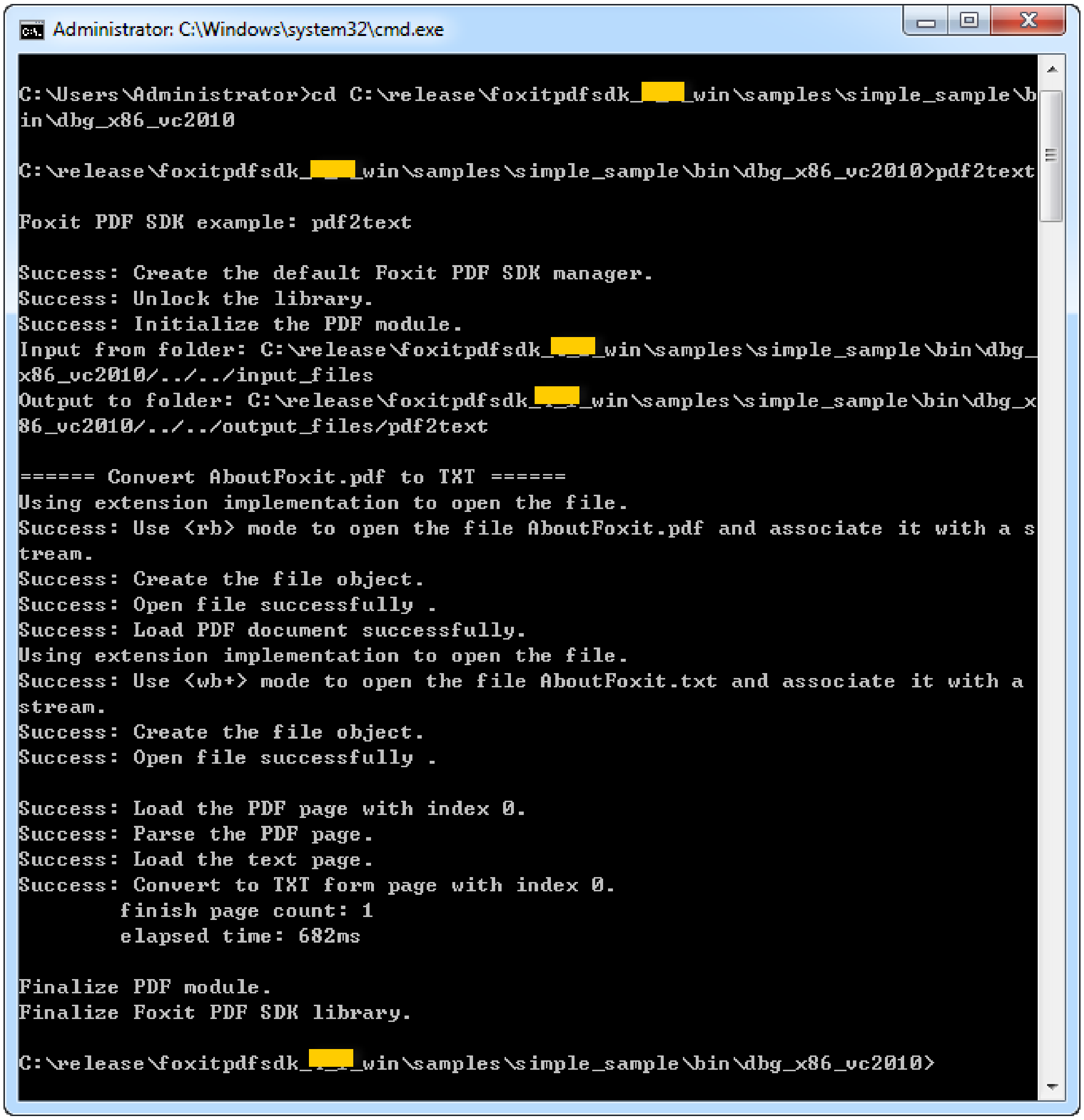
Assume that your Visual Studio version is 2010. And other demos also run in Visual Studio 2010. Double-click on “pdf2text_vc2010.vcxproj” under “samples/simple_samples/pdf2text” as shown in Figure 3-1, and then build it.



pdf2text.cpp
pdf2text.h
pdf2text_vc2008.vcproj
pdf2text_vc2010.vcxproj
pdf2text_vc2010.vcxproj.filters

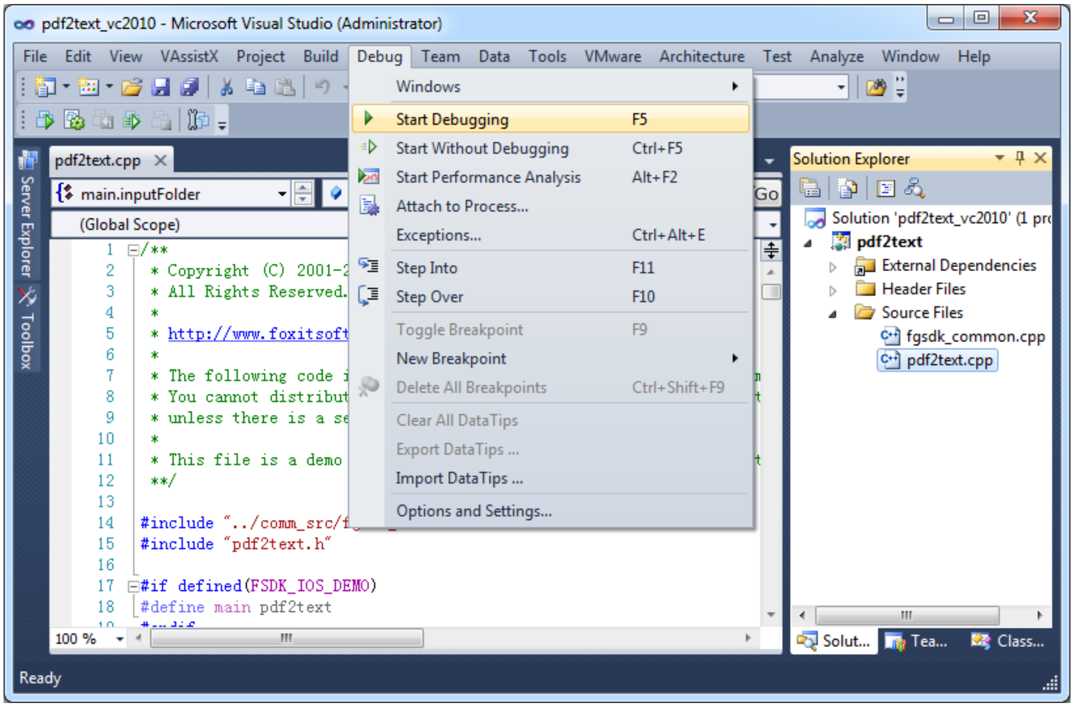
Figure 3-1

The executable file “pdf2text.exe” is generated in folder “bin\dbg_x86_vc2010”, which depends on the build configuration. There are two options to run executable file: by command line or Visual studio. When running in command line, start “cmd.exe”, navigate to “bin\dbg_x86_vc2010” and run “pdf2text.exe”. The output is shown in Figure 3-2. When running in Visual Studio, click on “Debug->Start Debugging” or “Debug->Start Without Debugging” on the menu bar to run pdf2text.exe, which is shown in Figure3-3.



```
C:\Users\Administrator>cd C:\release\foxitpdfsdk_[redacted]_win\samples\simple_sample\b
in\dbg_x86_vc2010

C:\release\foxitpdfsdk_[redacted]_win\samples\simple_sample\bin\dbg_x86_vc2010>pdf2text

Foxit PDF SDK example: pdf2text

Success: Create the default Foxit PDF SDK manager.
Success: Unlock the library.
Success: Initialize the PDF module.
Input from folder: C:\release\foxitpdfsdk_[redacted]_win\samples\simple_sample\bin\dbg_
x86_vc2010/../../input_files
Output to folder: C:\release\foxitpdfsdk_[redacted]_win\samples\simple_sample\bin\dbg_x
86_vc2010/../../output_files/pdf2text

====== Convert AboutFoxit.pdf to TXT ======
Using extension implementation to open the file.
Success: Use <rb> mode to open the file AboutFoxit.pdf and associate it with a s
tream.
Success: Create the file object.
Success: Open file successfully .
Success: Load PDF document successfully.
Using extension implementation to open the file.
Success: Use <wb+> mode to open the file AboutFoxit.txt and associate it with a
stream.
Success: Create the file object.
Success: Open file successfully .

Success: Load the PDF page with index 0.
Success: Parse the PDF page.
Success: Load the text page.
Success: Convert to TXT form page with index 0.
        finish page count: 1
        elapsed time: 682ms

Finalize PDF module.
Finalize Foxit PDF SDK library.

C:\release\foxitpdfsdk_[redacted]_win\samples\simple_sample\bin\dbg_x86_vc2010>
```

Figure 3-2

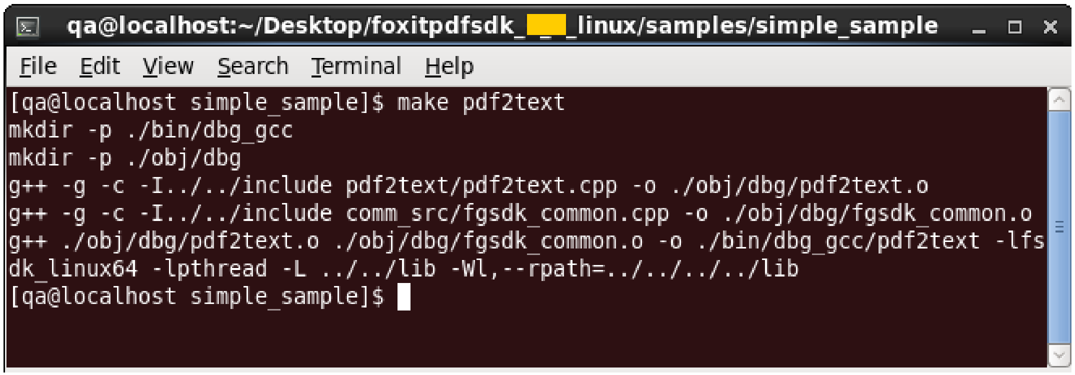


```
/**
 * Copyright (C) 2001-2
 * All Rights Reserved.
 *
 * http://www.foxitsoft
 *
 * The following code i
 * You cannot distribut
 * unless there is a se
 *
 * This file is a demo
 **/

#include "../comm_src/f
#include "pdf2text.h"

#if defined(FSDK_IOS_DEMO)
#define main pdf2text
```

Figure 3-3

For Linux

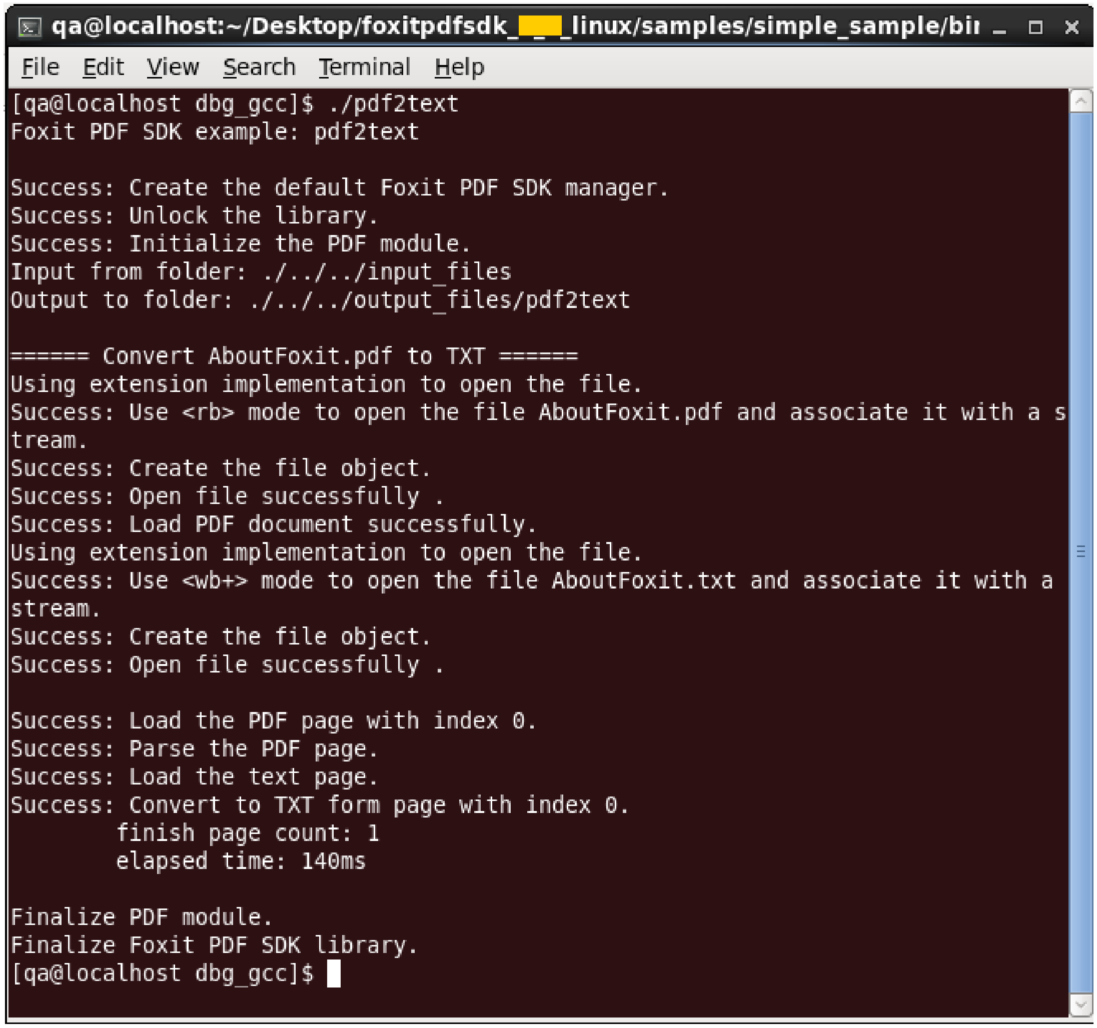
For Linux, demos should be compiled by using the make file in GCC compilers. In a terminal window, run "make pdf2text" to build the pdf2text demo which is shown in Figure 3-4.


qa@localhost:~/Desktop/foxitpdfsdk_[redacted]_linux/samples/simple_sample
File Edit View Search Terminal Help

```
[qa@localhost simple_sample]$ make pdf2text
mkdir -p ./bin/dbg_gcc
mkdir -p ./obj/dbg
g++ -g -c -I../../include pdf2text/pdf2text.cpp -o ./obj/dbg/pdf2text.o
g++ -g -c -I../../include comm_src/fgsdk_common.cpp -o ./obj/dbg/fgsdk_common.o
g++ ./obj/dbg/pdf2text.o ./obj/dbg/fgsdk_common.o -o ./bin/dbg_gcc/pdf2text -lfs
dk_linux64 -lpthread -L ../../lib -Wl,--rpath=../../../../lib
[qa@localhost simple_sample]$ 
```

Figure 3-4

After building, the binary files are generated in folder "samples/simple samples/bin/rel_gcc" or "samples/simple samples/bin/dbg_gcc" depending on the build option. Navigate to the folder with the terminal, and run the binary file to get the pdf2text demo running. The screenshot is shown in Figure 3-5.

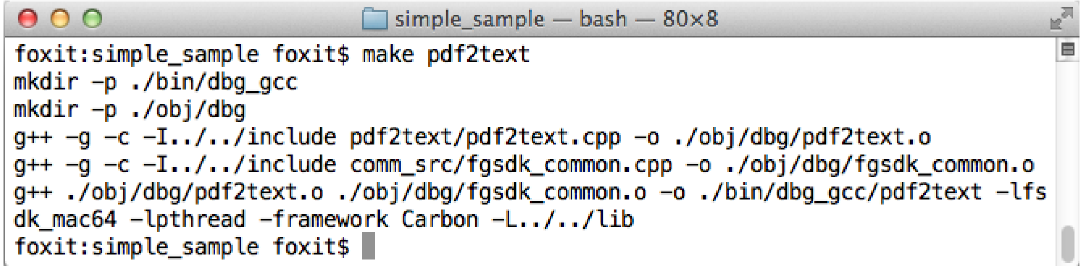

qa@localhost:~/Desktop/foxitpdfsdk_[redacted]_linux/samples/simple_sample/bi
File Edit View Search Terminal Help

```
[qa@localhost dbg_gcc]$ ./pdf2text
Foxit PDF SDK example: pdf2text

Success: Create the default Foxit PDF SDK manager.
Success: Unlock the library.
Success: Initialize the PDF module.
Input from folder: ./../../input_files
Output to folder: ./../../output_files/pdf2text

====== Convert AboutFoxit.pdf to TXT ======
Using extension implementation to open the file.
Success: Use <rb> mode to open the file AboutFoxit.pdf and associate it with a s
tream.
Success: Create the file object.
Success: Open file successfully .
Success: Load PDF document successfully.
Using extension implementation to open the file.
Success: Use <wb+> mode to open the file AboutFoxit.txt and associate it with a
stream.
Success: Create the file object.
Success: Open file successfully .

Success: Load the PDF page with index 0.
Success: Parse the PDF page.
Success: Load the text page.
Success: Convert to TXT form page with index 0.
        finish page count: 1
        elapsed time: 140ms

Finalize PDF module.
Finalize Foxit PDF SDK library.
[qa@localhost dbg_gcc]$ 
```

Figure 3-5

For Mac

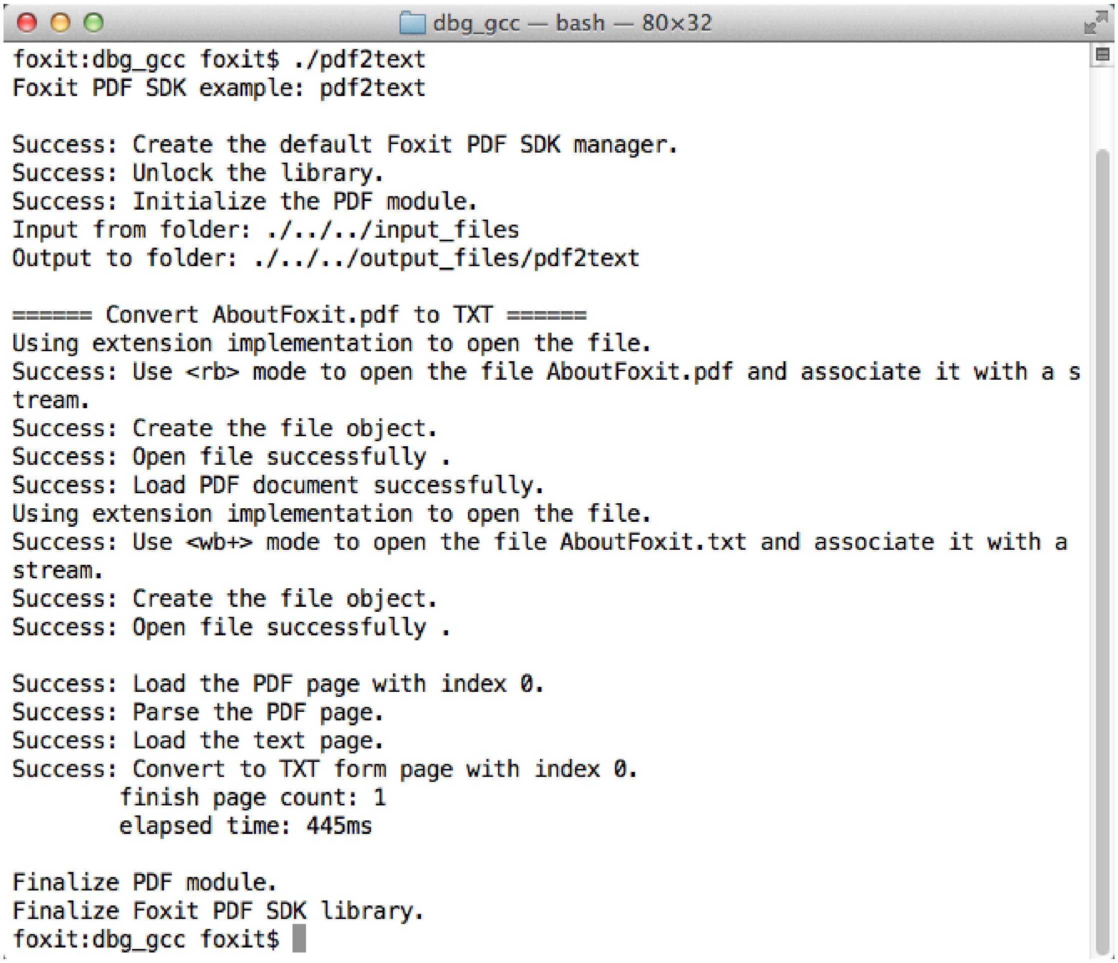
For Mac, demos also should be compiled by using the make file in GCC compilers. In a terminal window, run “make pdf2text” to build the pdf2text demo which is shown in Figure 3-6.



```
foxit:simple_sample foxit$ make pdf2text
mkdir -p ./bin/dbg_gcc
mkdir -p ./obj/dbg
g++ -g -c -I../../include pdf2text/pdf2text.cpp -o ./obj/dbg/pdf2text.o
g++ -g -c -I../../include comm_src/fgsdk_common.cpp -o ./obj/dbg/fgsdk_common.o
g++ ./obj/dbg/pdf2text.o ./obj/dbg/fgsdk_common.o -o ./bin/dbg_gcc/pdf2text -lfs
dk_mac64 -lpthread -framework Carbon -L../../lib
foxit:simple_sample foxit$ 
```

Figure 3-6

After building, the binary files are generated in folder “samples/simple samples/bin/rel_gcc” or "samples/simple samples/bin/dbg_gcc" depending on the build option. Navigate to the folder with the terminal, and run the binary file to get the pdf2text demo running. The screenshot is shown in Figure 3-7.



```
foxit:dbg_gcc foxit$ ./pdf2text
Foxit PDF SDK example: pdf2text

Success: Create the default Foxit PDF SDK manager.
Success: Unlock the library.
Success: Initialize the PDF module.
Input from folder: ./../../input_files
Output to folder: ./../../output_files/pdf2text

====== Convert AboutFoxit.pdf to TXT ======
Using extension implementation to open the file.
Success: Use <rb> mode to open the file AboutFoxit.pdf and associate it with a s
tream.
Success: Create the file object.
Success: Open file successfully .
Success: Load PDF document successfully.
Using extension implementation to open the file.
Success: Use <wb+> mode to open the file AboutFoxit.txt and associate it with a
stream.
Success: Create the file object.
Success: Open file successfully .

Success: Load the PDF page with index 0.
Success: Parse the PDF page.
Success: Load the text page.
Success: Convert to TXT form page with index 0.
        finish page count: 1
        elapsed time: 445ms

Finalize PDF module.
Finalize Foxit PDF SDK library.
foxit:dbg_gcc foxit$
```

Figure 3-7

For iOS

For iOS, demos should be built and run in Xcode. Load the Xcode solution files "simple_sample_ios.xcodeproj" under "samples/simple_sample/simple_sample_ios" folder which is shown in Figure 3-8. Here the Xcode version is 5.0.

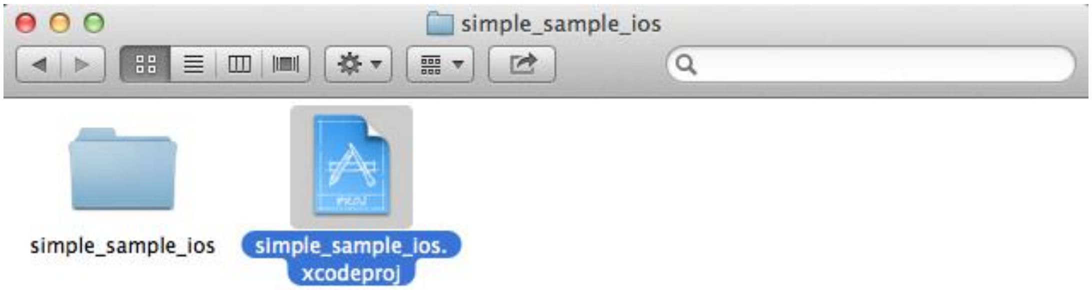
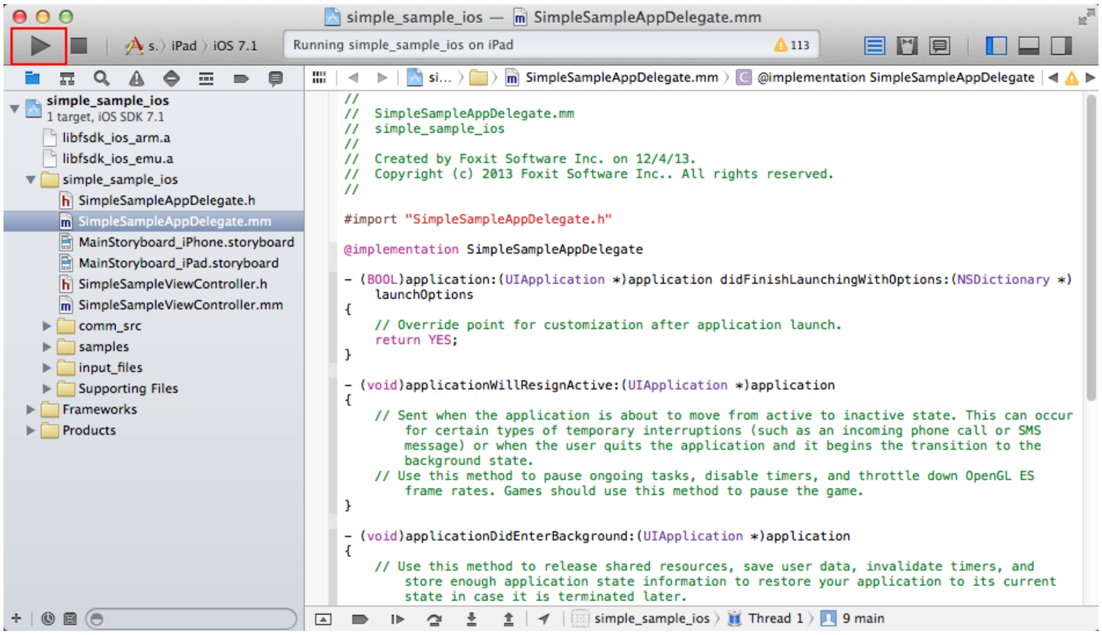


Figure 3-8

After loading the solution, click on “Run” on the menu bar to build the solution. A screenshot is shown in Figure 3-9. The output files for all demos are put in “**sandbox**”.


simple_sample_ios — SimpleSampleAppDelegate.mm
s. 〉 iPad 〉 iOS 7.1
Running simple_sample_ios on iPad
113
si... 〉 〉 SimpleSampleAppDelegate.mm 〉 @implementation SimpleSampleAppDelegate
simple_sample_ios
1 target, iOS SDK 7.1
libfsdk_ios_arm.a
libfsdk_ios_emu.a
simple_sample_ios
SimpleSampleAppDelegate.h
SimpleSampleAppDelegate.mm
MainStoryboard_iPhone.storyboard
MainStoryboard_iPad.storyboard
SimpleSampleViewController.h
SimpleSampleViewController.mm
comm_src
samples
input_files
Supporting Files
Frameworks
Products
simple_sample_ios 〉 Thread 1 〉 9 main

```
//
//  SimpleSampleAppDelegate.mm
//  simple_sample_ios
//
//  Created by Foxit Software Inc. on 12/4/13.
//  Copyright (c) 2013 Foxit Software Inc.. All rights reserved.
//

#import "SimpleSampleAppDelegate.h"

@implementation SimpleSampleAppDelegate

- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)
    launchOptions
{
    // Override point for customization after application launch.
    return YES;
}

- (void)applicationWillResignActive:(UIApplication *)application
{
    // Sent when the application is about to move from active to inactive state. This can occur
        for certain types of temporary interruptions (such as an incoming phone call or SMS
        message) or when the user quits the application and it begins the transition to the
        background state.
    // Use this method to pause ongoing tasks, disable timers, and throttle down OpenGL ES
        frame rates. Games should use this method to pause the game.
}

- (void)applicationDidEnterBackground:(UIApplication *)application
{
    // Use this method to release shared resources, save user data, invalidate timers, and
        store enough application state information to restore your application to its current
        state in case it is terminated later.
```

Figure 3-9

After Building the solution successfully, the iOS simulator will be started as shown in Figure 3-10, if you choose to run in a simulator. In this tutorial, we select the iPad/iOS 7.0 simulator.

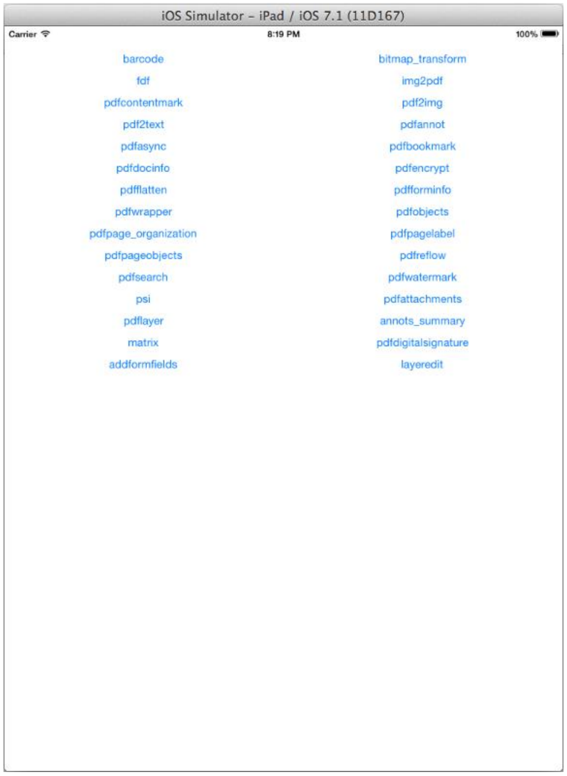
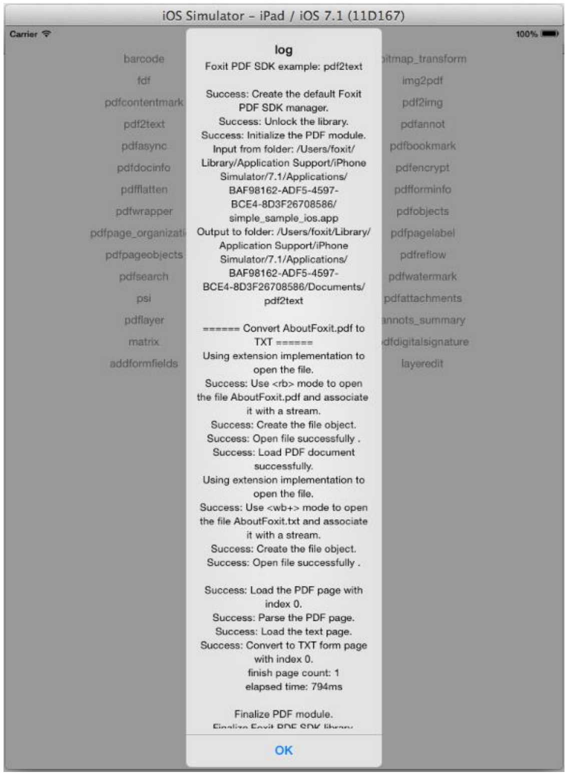


Figure 3-10

All the demo functionalities are shown in Figure 3-10. In this demo, click on “pdf2text”, the log information is shown in Figure 3-11.



```
Foxit PDF SDK example: pdf2text

Success: Create the default Foxit
PDF SDK manager.
Success: Unlock the library.
Success: Initialize the PDF module.
Input from folder: /Users/foxit/
Library/Application Support/iPhone
Simulator/7.1/Applications/
BAF98162-ADF5-4597-
BCE4-8D3F26708586/
simple_sample_ios.app
Output to folder: /Users/foxit/Library/
Application Support/iPhone
Simulator/7.1/Applications/
BAF98162-ADF5-4597-
BCE4-8D3F26708586/Documents/
pdf2text

====== Convert AboutFoxit.pdf to
TXT ======
Using extension implementation to
open the file.
Success: Use <rb> mode to open
the file AboutFoxit.pdf and associate
it with a stream.
Success: Create the file object.
Success: Open file successfully .
Success: Load PDF document
successfully.
Using extension implementation to
open the file.
Success: Use <wb+> mode to open
the file AboutFoxit.txt and associate
it with a stream.
Success: Create the file object.
Success: Open file successfully .

Success: Load the PDF page with
index 0.
Success: Parse the PDF page.
Success: Load the text page.
Success: Convert to TXT form page
with index 0.
finish page count: 1
elapsed time: 794ms

Finalize PDF module.
Finalize Foxit PDF SDK library.
```

Figure 3-11

3.2 Bitmap_Transform

3.2.1 Description

1) Purpose: illustrates how to use Foxit PDF SDK to load different types of image files and convert image files by flipping, stretching and transforming.
2) Property: this demo is a console application without UI.

   Usage: *bitmap_tranform*
3) Process: this demo loads specific image files from input directory. Then it will convert images to bitmaps and do transformation on bitmaps by flipping, stretching and transforming etc., and save changed bitmaps to output directory. Once completed, the demo will exit. All the information about execution process will be recorded into log file.
4) Input directory: used to place input files.

   Default: /samples/simple_sample/input_files. Here're two ways:
   a) Use default files: FoxitCorporation.bmp.
   b) Set default sub-folder: create a sub-folder named "bitmap_transform" and put all input files into this sub-folder. When running, this demo will use files in sub-folder as input files.
5) Output directory: use to store output files, including log.txt.

   Default: /samples/simple_sample/output_files/bitmap_transform.
6) Program files

   bitmap_transform.cpp: source code of loading images and converting these files.

3.2.2 SDK functions

The demo of bitmap_transform uses following additional functions of Foxit PDF SDK:

1) FSCRT_ImageFile_Create: used for creating an image file object.
2) FSCRT_ImageFile_AddFrame: used for adding a frame image.
3) FSCRT_ImageFile_Release: used for releasing an image file object.
4) FSPDF_Doc_StartSaveToFile: used for starting a step-by-step saving progress for PDF files.
5) FSCRT_Image_LoadFromFile: used for loading image files.
6) FSCRT_Image_CountFrams: used for getting the number of frames of image files.
7) FSCRT_Image_LoadFrame: used for loading a frame of a given image file.

8) FSCRT_Image_GetCurrentFrameBitmap: used for getting the bitmap of the current frame.
9) FSCRT_Bitmap_GetFlipped: used for flipping a bitmap.
10) FSCRT_Bitmap_GetSize: used for getting the width and height of a given bitmap.
11) FSCRT_Bitmap_GetFormat: used for getting the format of a given bitmap.
12) FSCRT_Bitmap_Create: used for creating a bitmap.
13) FSCRT_Bitmap_FillRect: used for filling a rectangle of a given bitmap with a certain color.
14) FSCRT_Bitmap_StretchTo: used for stretching a bitmap.
15) FSCRT_Bitmap_TransformTo: used for converting a bitmap by a given matrix.
16) FSCRT_Bitmap_CalcBBox: used for calculating the rectangle borders of a given bitmap.
17) FSCRT_Bitmap_ConvertFormat: used for converting the format of a given bitmap.
18) FSCRT_Bitmap_Release: used for releasing a bitmap object.
19) FSCRT_Image_Release: used for releasing an image object.

3.2.3 Run

Please refer to the 3.1.3 to run this demo.

3.3 Barcode

3.3.1 Description

1) Purpose: illustrates how to generate different encoding formats of barcodes and save them as bitmap image files.
2) Property: this demo is a console application without UI.

   Usage: *barcode*
3) Process: this demo can generate 8 types of barcodes from a given character string and save them as bitmap files whose names are corresponded with their encoding formats. These bitmap files will be generated to output directory. Once completed, the demo will exit. All the information about execution process will be recorded into log file.
4) Input directory: no input file is needed.
5) Output directory: used to store output files, including log.txt.
   a) Default: /samples/simple_sample/output_files/barcode.
   b) User-defined: set path of output folder through command line argument and output directory will be "/outputfolder/barcode".
6) Program files

   barcode.cpp: source code of generating barcode image files in different encoding formats.

3.3.2 SDK functions

The demo of barcode uses following additional functions of Foxit PDF SDK:

1) FSCRT_ImageFile_Create: used for creating an image file object.
2) FSCRT_ImageFile_AddFrame: used for adding a frame image.
3) FSCRT_ImageFile_Release: used for releasing an image file object.
4) FSPDF_Doc_StartSaveToFile: used for starting a step-by-step saving progress for PDF files.
5) FSCRT_BCModule_Initialize: used for initializing barcode modules.
6) FSCRT_BCModule_Finalize: used for releasing barcode modules.
7) FSCRT_Barcode_GenerateBitmap: used for generating bitmaps with the corresponding barcode formats.
8) FSCRT_Bitmap_Release: used for releasing a bitmap object.

3.3.3 Run

Please refer to the 3.1.3 to run this demo.

3.4 Pdfdocinfo

3.4.1 Description

1) Purpose: illustrates how to use Foxit PDF SDK to get and set attributes of Viewer Preference and to access the metadata of a PDF file.

2) Property: this demo is a console application without UI.

   It supports command line argument.

   Usage: *pdfdocinfo [<srcfile> /o [destfolder]]*

<srcfile>	Path of a PDF file as input file.
/o [destfolder]	Path of a folder as output directory. If not set, use default output directory.

3) Process: this demo loads PDF files from input directory. Then for each input PDF file, it will get and set the attributes of Viewer Preference and metadata, and generated PDF files will be stored into output directory. Once completed, the demo will exit. All the information about execution process will be recorded into log file.

4) Input directory: used to place input files.
    a) Default: /samples/simple_sample/input_files. Here're two way:
        i. Use default files: PageLabels.pdf.
        ii. Set default sub-folder: create a sub-folder named "pdfdocinfo" and put all input files into this sub-folder. When running, this demo will use files in sub-folder as input files.
    b) User-defined: set single input file through command line argument.

5) Output directory: used to store output files, including log.txt.
    a) Default: /samples/simple_sample/output_files/pdfdocinfo.
    b) User-defined: set path of output folder through command line argument and output directory will be "/outputfolder/pdfdocinfo".

6) Program files:

   pdfdocinfo.cpp: source code of setting and getting attributes of Viewer Preference and accessing the metadata of PDF files.

3.4.2 SDK functions

The demo of pdfdocinfo uses following additional functions of Foxit PDF SDK:

1) FSPDF_Doc_StartSaveToFile: used for starting an incremental saving progress for PDF files.
2) FSPDF_Doc_StartLoad: used for loading a PDF file.
3) FSPDF_Doc_CountPages: used for counting the number of pages of a PDF file.
4) FSPDF_Doc_GetPage: used for getting a page object of a PDF file.
5) FSPDF_Page_Clear: used for clearing a page object of a PDF file and releasing page resource without destroying the page object.
6) FSPDF_Doc_Close: used for closing a given PDF file.
7) FSPDF_ViewerPref_SetUIVisibility: used for setting user interface display status in Viewer Preference.
8) FSPDF_ViewerPref_SetNonFullScreenPageMode: used for setting the display mode in Viewer Preference.
9) FSPDF_ViewerPref_SetReadingDirection: used for setting the reading direction in Viewer Preference.
10) FSPDF_ViewerPref_SetAreaType: used for setting the area type in Viewer Preference.
11) FSPDF_ViewerPref_SetPrintScale: used for setting the print scale in Viewer Preference.
12) FSPDF_ViewerPref_SetPrintCopies: used for setting the number of copies to be printed in Viewer Preference.
13) FSPDF_ViewerPref_SetPrintRanges: used for setting the print range in Viewer Preference.
14) FSPDF_ViewerPref_GetUIVisibility: used for getting the user interface display status in Viewer Preference.
15) FSPDF_ViewerPref_GetNonFullScreenPageMode: used for getting the display mode in Viewer Preference.
16) FSPDF_ViewerPref_GetReadingDirection: used for getting the reading direction in Viewer Preference.
17) FSPDF_ViewerPref_GetAreaType: used for getting the area type in Viewer Preference.
18) FSPDF_ViewerPref_GetPrintScale: used for getting the print scale in Viewer Preference.
19) FSPDF_ViewerPref_GetPrintScale: used for getting the number of copies to be printed in Viewer Preference.
20) FSPDF_ViewerPref_GetPrintRanges: used for getting the print range in Viewer Preference.
21) FSPDF_Metadata_SetString: used for setting the value of attributes in metadata.

22) FSPDF_Metadata_SetDateTime: used for setting PDF creation and modification date & time in metadata.
23) FSPDF_Metadata_GetString: used for getting the value of attribute in metadata.
24) FSPDF_Metadata_GetDateTime: used for getting the PDF creation and modification date & time in metadata.

3.4.3 Run

Please refer to the 3.1.3 to run this demo.

3.5 Pdfbookmark

3.5.1 Description

1) Purpose: illustrates how to use Foxit PDF SDK to enumerate and modify bookmarks of PDF files.
2) Property: this demo is a console application without UI.

   It supports command line argument.

   Usage: *pdfbookmark [<srcfile> /o [destfolder]]*

<srcfile>	Path of a PDF file as input file.
/o [destfolder]	Path of a folder as output directory. If not set, use default output directory.

3) Process: this demo loads specified PDF files from input directory. For each PDF file, it will get information of all bookmarks'. Then it will modify bookmarks by creating a child bookmark for them, editing data and deleting data. Modified PDFs will be stored into output directory, with "_add", "_edit" or "_delete" as suffix of file name. All the information about execution process will be recorded into log file.
4) Input directory: used to place input files.
   a) Default: /samples/simple_sample/input_files. Here're two ways:
      i. Use default files: Bookmark.pdf.
      ii. Set default sub-folder: create a sub-folder named "pdfbookmark" and put all input files into this sub-folder. When running, this demo will use files in sub-folder as input files.
   b) User-defined: set single input file through command line argument.
5) Output directory: used to store output files, including log.txt.
   a) Default: /samples/simple_sample/output_files/pdfbookmark.
   b) User-defined: set path of output folder through command line argument and output directory will be "/outputfolder/pdfbookmark".
6) Program files:

   pdfbookmark.cpp: source code of enumerating, modifying the bookmarks of PDF files.

3.5.2 SDK functions

The demo of pdfbookmark uses following additional functions of Foxit PDF SDK:

1) FSPDF_Doc_StartSaveToFile: used for starting an incremental saving progress for PDF

files.

2) FSPDF_Doc_Close: used for closing a given PDF document.
3) FSPDF_BookmarkData_Init: used for initializing property data of bookmarks.
4) FSPDF_BookmarkData_Clear: used for clearing property data of bookmarks.
5) FSPDF_Bookmark_CreateIterator: used for creating iterators of bookmarks.
6) FSPDF_Bookmark_ReleaseIterator: used for releasing iterators of bookmarks.
7) FSPDF_Bookmark_MoveToRoot: used for moving to root nodes of bookmarks.
8) FSPDF_Bookmark_MoveToFirstChild: used for moving to the first child node.
9) FSPDF_Bookmark_MoveToNextSibling: used for moving to the next sibling node.
10) FSPDF_Bookmark_MoveToParent: used for moving to the parent node.
11) FSPDF_Bookmark_Insert: used for inserting a new bookmark.
12) FSPDF_Bookmark_SetData: used for setting the data of a bookmark.
13) FSPDF_Bookmark_Remove: used for deleting a bookmark.

3.5.3 Run

Please refer to the 3.1.3 to run this demo.

3.6 Pdfannot

3.6.1 Description

1) Purpose: illustrates how to use Foxit PDF SDK to add multiple types of annotations to PDF files.

2) Property: this demo is a console application without UI.

   It supports command line argument.

   Usage: *pdfannot [<srcfile> /r [pagerange] /o [destfolder]]*

<srcfile>	Path of a PDF file as input file
/r [pagerange]	Page range, used to specify indexes of PDF pages in input PDF file, such as "0", "1-3,5". Page index starts from 0 and should be valid within page count. Invalid index will be ignored. If input page range is totally invalid, page range will be set to first page by default.
/o [destfolder]	Path of a folder as output directory. If not set, use default output directory.

3) Process: this demo loads specified PDF files from input directory. For each PDF file, this demo will add multiple types of annotations into specific pages (Default: first page) and then save the changed PDF as a new one to output directory. Once completed, the demo will exit. All the information about execution process will be recorded into log file.

4) Input directory: used to place input files. Specially, please ensure that “FoxitLogo.jpg” is always available in input directory.
   a) Default: /samples/simple_sample/input_files. Here’re two ways:
      i. Use default files: AboutFoxit.pdf.
      ii. Set default sub-folder: create a sub-folder named “pdfannot” and put all input files into this sub-folder. When running, this demo will use files in sub-folder as input files.
   b) User-defined: set single input file through command line argument.

5) Output directory: use to store output files, including log.txt.
   a) Default: /samples/simple_sample/output_files/pdfannot.
   b) User-defined: set path of output folder through command line argument and output directory will be “/outputfolder/pdfannot”.

6) Program file：

pdfannot.cpp:	source code for adding annotations to PDF files.

3.6.2 SDK functions

The demo of pdfannot uses following additional functions of Foxit PDF SDK:

1) FSCRT_PDFModule_Initialize: used for initializing a PDF module.
2) FSCRT_PDFModule_Finalize: used for releasing a PDF module.
3) FSPDF_Doc_StartSaveToFile: used for starting an incremental saving progress for PDF files.
4) FSPDF_Doc_GetPage: used for getting a PDF page object.
5) FSPDF_Page_StartParse: used for starting to parse a PDF page object and returning a progress object.
6) FSPDF_Annot_Add: used for adding an annotation object into a PDF page. It specifies a rectangular area, a type, an index and an annotation type filter string as parameters and then creates annotations and adds them into to a given PDF page.
7) FSPDF_Annot_AddState: used for creating and adding a state annotation object to an annotation object by setting state models and states.
8) FSPDF_Annot_Remove: used for removing an annotation object from a given PDF page.
9) FSPDF_Annot_Move: used for resetting the rectangular area of an annotation object.
10) FSPDF_Annot_ResetAppearance: used for resetting appearance streams of annotation objects.
11) FSPDF_Annot_SetFlags: used for setting flag bits of annotation objects.
12) FSPDF_Annot_SetName: used for setting names of annotation objects.
13) FSPDF_Annot_SetSubject: used for setting subjects of annotation objects.
14) FSPDF_Annot_SetTitle: used for setting titles of annotation objects.
15) FSPDF_Annot_SetContents: used for setting contents of annotation objects.
16) FSPDF_Annot_SetIconName: used for setting icon names of annotation objects.
17) FSPDF_Annot_SetCreationDateTime: used for setting the creation time of annotation objects.
18) FSPDF_Annot_SetModifiedDateTime: used for setting the modified time of annotation objects.
19) FSPDF_Annot_SetBorder: used for setting the border properties of annotation objects.
20) FSPDF_Annot_SetQuadPoints: used for setting the quadrangle linked lists of annotation objects.

21) FSPDF_Annot_SetColor: used for setting the colors of annotation objects.
22) FSPDF_Annot_SetOpacity: used for setting the opacity of annotation objects.
23) FSPDF_Annot_SetLineEndingStyles: used for setting the line styles of annotation objects.
24) FSPDF_Annot_SetLinePosition: used for setting the positions of line annotation objects.
25) FSPDF_Annot_SetLeaderLineLength: used for setting the leader line length of line annotation objects.
26) FSPDF_Annot_SetLeaderLineExtension: used for setting the leader line extension length of line annotation objects.
27) FSPDF_Annot_SetLeaderLineOffset: used for setting the leader line offset distance of line annotation objects.
28) FSPDF_Annot_SetCaptionContents: used for setting the caption contents to enable or disable line annotation objects.
29) FSPDF_Annot_SetCaptionPosition: used for setting the caption content position of line annotation objects.
30) FSPDF_Annot_SetCaptionOffset: used for setting the offset distance of caption content of line annotation objects compared with lines.
31) FSPDF_Annot_SetVertices: used for setting the vertical linked list of polyline and polygon annotation objects.
32) FSPDF_Annot_SetAlignment: used for setting the alignment of text annotation objects.
33) FSPDF_Annot_SetDefaultAppearance: used for setting the default appearance of text annotation objects.
34) FSPDF_Annot_SetHighlightingMode: used for setting the highlighting mode of annotation objects.
35) FSPDF_Annot_InsertAction: used for setting the action response of annotation objects.
36) FSPDF_Annot_SetInkList: used for setting the track point linked list of drawing annotation objects.
37) FSCRT_PathData_Create: used for creating path data objects.
38) FSCRT_PathData_AddPointsCount: used for setting point count information of path data objects.
39) FSCRT_PathData_SetPoint: used for setting the point information of path data objects.
40) FSCRT_PathData_Clear: used for clearing the cached data of path data objects.
41) FSCRT_Image_LoadFromFile: used for loading images from a given image file.

42) FSCRT_Image_Release: used for releasing an image object.
43) FSPDF_Annot_SetStamp: used for setting an image to a stamp annotation.
44) FSPDF_Page_Clear: used for clearing a PDF page object and releasing page resources but without destroying the page object.
45) FSPDF_Doc_Close: used for closing a given PDF document.

3.6.3 Run

Please refer to the 3.1.3 to run this demo.

3.6.4 Note

Types of added annotations covers most of what Foxit PDF SDK supports to add.

3.7 Pdfforminfo

3.7.1 Description

1) Purpose: illustrates how to use Foxit PDF SDK to obtain form information, import form data from an FDF file to a PDF file and export forms data from a PDF file to an FDF file.

2) Property: the demo program is a console application without UI.

   Usage: *pdfforminfo*

3) Process: the demo loads specific PDFs and FDFs from input directory. For each FDF file, the demo will open the original PDF from the input directory and then import the form data from FDF to the original PDF. For each PDF file, this demo will generate a new FDF file and export the PDF's form data to it. Once completed, the demo will exit. All the information about execution process will be recorded into log file.

4) Input directory: used to place input files.

   Default: /samples/simple_sample/input_files. Here're two ways:

   i. Use default files: InformationCollectionData.fdf, InformationCollection.pdf.

   ii. Set default sub-folder: create a sub-folder named "pdfforminfo" and put all input files into this sub-folder. When running, this demo will enumerate files in sub-folder as input files

5) Output directory: used to store output files, including log.txt.

   Default: /samples/simple_sample/output_files/pdfforminfo.

6) Program files:

   pdfforminfo.cpp: source code of importing form data from FDF files to PDF files, exporting form data from PDF files to FDF files, and obtaining form data.

3.7.2 SDK functions

The demo of pdfforminfo uses following additional functions of Foxit PDF SDK:

1) FSCRT_PDFModule_Initialize: used for initializing a PDF module.
2) FSCRT_PDFModule_Finalize: used for releasing a PDF module.
3) FSPDF_Doc_StartSaveToFile: used for starting an incremental saving progress for PDF files.
4) FSPDF_Doc_LoadFromFile: used for loading PDF files.

5) FSPDF_Doc_CountPages: used for counting pages of a PDF file.
6) FSPDF_Doc_GetPage: used for getting a page object of a PDF file.
7) FSPDF_Page_Clear: used for clearing a page object of a PDF file and releasing the page resource without destroying the page object.
8) FSPDF_Doc_Close: used for closing PDF files.
9) FSFDF_Doc_Load: used for loading FDF files.
10) FSFDF_Doc_Create: used for creating FDF files.
11) FSFDF_Doc_GetPDFPath: used for getting the path of corresponding PDF file.
12) FSFDF_Doc_Close: used for closing FDF files.
13) FSFDF_Doc_Save: used for saving FDF files.
14) FSPDF_Form_LoadFromPDF: used for loading forms in a PDF file.
15) FSPDF_Form_ImportFromFDFDoc: used for importing form data from an FDF file.
16) FSPDF_Form_Release: used for releasing forms.
17) FSPDF_Form_CountFields: used for counting form fields.
18) FSPDF_Form_GetField: used for getting a form field and the category of forms based on the index.
19) FSPDF_Form_ExportToFDFDoc: used for exporting form data to FDF file.
20) FSPDF_FormField_GetAlignment: used for getting alignment of form based on form field.
21) FSPDF_FormField_GetValue: used for getting the values of a form field.
22) FSPDF_FormField_GetDefaultValue: used for getting the default value of form based on form field.
23) FSPDF_Page_LoadAnnots: used for loading annotations in a specified page.
24) FSPDF_Page_UnloadAnnots: used for unloading annotations in a specified page.
25) FSPDF_FormField_CountControls: used for counting controls in a form field.

3.7.3 Run

Please refer to the 3.1.3 to run this demo.

3.8 Pdfpagelabel

3.8.1 Description

1) Purpose: illustrates how to use Foxit PDF SDK to get page label information from PDF files.
2) Property: this demo is a console application without UI.

   It supports command line argument.

   Usage: pdfpagelabel *[* <srcfile> */o [*destfolder*]]*

<srcfile>	Path of a PDF file as input file.
/o [destfolder]	Path of a folder as output directory. If not set, use default output directory.

3) Process: the demo loads specified PDF files from input directory. For each PDF file, it will get page indexes and then get page label information based on page indexes. Once completed, the demo will exit. All the information about execution process will be recorded into log file.
4) Input directory: used to place input files.
   a) Default: /samples/simple_sample/input_files. Here're two way:
      i. Use default files: PageLabels.pdf.
      ii. Set default sub-folder: create a sub-folder named "pdfpagelabel" and put all input files into this sub-folder. When running, this demo will use files in sub-folder as input files.
   b) User-defined: set single input file through command line argument.
5) Output directory: used to store output files, including log.txt.
   a) Default: /samples/simple_sample/output_files/pdfpagelabel.
   b) User-defined: set path of output folder through command line argument and output directory will be "/outputfolder/pdfpagelabel".
6) Program files:

   pdfpagelabel.cpp: source code of getting page label by page's index.

3.8.2 SDK functions

The demo of pdfpagelabel uses following additional functions of Foxit PDF SDK:

1) FSPDF_Doc_StartSaveToFile: used for starting an incremental saving progress for PDF

files.

2) FSPDF_Doc_StartLoad: used for loading a PDF file.
3) FSPDF_Doc_CountPages: used for counting the number of pages of a PDF file.
4) FSPDF_Doc_GetPage: used for getting a page object of PDF file.
5) FSPDF_Page_Clear: used for clearing a page object of PDF file without destroying the page object.
6) FSPDF_Doc_Close: used for closing PDF files.
7) FSPDF_Doc_PageIndexToPageLabel: used for getting page label information by page.

3.8.3 Run

Please refer to the 3.1.3 to run this demo.

3.9 Pdfpage_organization

3.9.1 Description

1) Purpose: illustrates how to use Foxit PDF SDK to split, merge PDF files, delete a PDF page and set a page index.
2) Property: this demo is a console application without UI.

   Usage: *pdfpage_organization*
3) Process: this demo loads specific PDF files from input directory. This demo will insert some pages into a PDF file and save as a new PDF file, and merge two PDF files into one. In addition, this demo can delete a page from given PDF file and set page index for some pages. Once completed, the demo will exit. All the information about execution process will be recorded into log file.
4) Input directory:use to place input files. Specially, please ensure that "InsertPages.pdf" is always available in input directory.

   Default: /samples/simple_sample/input_files. Here're two way:

   i. Use default files: SourcePage.pdf, InsertPages.pdf.
   ii. Set default sub-folder: create a sub-folder named "pdfpage_organization" and put all input files into this sub-folder. When running, this demo will enumerate files in sub-folder as input files
5) Output directory: used to store output files, including log.txt.

   Default: /samples/simple_sample/output_files/pdfpage_organization.
6) Program files:

   pdfpage_organization.cpp: source code of inserting pages into a PDF file, merging two PDF files, deleting pages and setting page's index.

3.9.2 SDK functions

The demo of pdfpage_organization uses following additional functions of Foxit PDF SDK:

1) FSPDF_Doc_StartSaveToFile: used for starting an incremental saving progress for PDF files.
2) FSPDF_Doc_LoadFromFile: used for loading a given PDF file.
3) FSPDF_Doc_CountPages: used for counting the pages of a PDF file.
4) FSPDF_Doc_GetPage: used for getting a page object of a PDF file.
5) FSPDF_Page_Clear: used for clearing a page object of PDF file and releasing the page resource without destroying it.

6) FSPDF_Doc_Close: used for closing a given PDF file.
7) FSPDF_Doc_Create: used for creating a new PDF document.
8) FSPDF_Doc_StartImportPages: used for starting a step-by-step process to import pages to document.
9) FSPDF_Page_Delete: used for deleting a page from a given PDF file.
10) FSPDF_Page_SetIndex: used for setting a page index.

3.9.3 Run

Please refer to the 3.1.3 to run this demo.

3.10 Pdfpageobjects

3.10.1 Description

1) Purpose: illustrates how to use Foxit PDF SDK to create a PDF, insert four supported page objects into the created PDF file, and save the file with inserted page objects.
2) Property: this demo is a console application without UI.

   Usage: *pdfpageobjects*
3) Process: this demo loads specified image files from input directory. This demo will create a new PDF file with a blank page, and insert four types of page objects to the new page. Then it will save the PDF to output directory. Once completed, the demo will exit. All the information about execution process will be recorded into log file.
4) Input directory: used to place input files.

   Default: /samples/simple_sample/input_files. Here're two ways:
   i. Use default files: FoxitCorporation.bmp.
   ii. Set default sub-folder: create a sub-folder named "pdfpageobjects" and put all input files into this sub-folder. When running, this demo will enumerate files in sub-folder as input files.
5) Output directory: used to store output files, including log.txt.

   Default: /samples/simple_sample/output_files/pdfpageobjects.
6) Program files:

   pdfpageobjects.cpp: source code of creating a PDF file, creating pages, inserting page objects and saving a PDF file.

3.10.2 SDK functions

The demo of pdfpageobjects uses following additional functions of Foxit PDF SDK:

1) FSPDF_Doc_Create: used for creating a PDF file.
2) FSPDF_Doc_Close: used for closing a given PDF file.
3) FSPDF_Doc_StartSaveToFile: used for starting an incremental saving progress for PDF files.
4) FSCRT_PathData_Create: used for creating a new graphic data object.
5) FSCRT_PathData_MoveTo: used for adding a point to a given graphic data object (as the start point).

6) FSCRT_PathData_LineTo: used for adding a point to a given graphic data object so as to form a line.
7) FSCRT_PathData_Release: used for releasing a graphic data object.
8) FSPDF_PageObject_SetMatrix: used for setting a matrix of a page object.
9) FSPDF_PageObjects_InsertObject: used for inserting a page object.
10) FSPDF_PageObjects_GenerateContents: used for generating page content of PDF file.
11) FSPDF_Page_Create: used for creating a new page.
12) FSPDF_Page_SetSize: used for setting the size of page.
13) FSPDF_Page_StartParse: used for parsing a PDF page.
14) FSPDF_Page_GetPageObjects: used for getting page objects in a given page.
15) FSPDF_Page_Clear: used for releasing a PDF page object without destroying it.
16) FSCRT_Font_Create: used for creating a font object (including attribute information of the font).
17) FSCRT_Font_Release: used for releasing font objects.
18) FSPDF_TextObject_Create: used for creating a text object.
19) FSPDF_TextObject_SetTextState: used for setting text state of a text object.
20) FSPDF_TextObject_SetUnicodeString: used for setting text information for a text object.
21) FSCRT_Image_LoadFromFile: used for loading images from image files.
22) FSCRT_Image_CountFrames: used for counting the frame numbers of an image.
23) FSCRT_Image_LoadFrame: used for loading a frame of an image.
24) FSCRT_Image_Release: used for releasing a given image.
25) FSPDF_ImageObject_Create: used for creating an image object.
26) FSPDF_ImageObject_SetImage: used for setting image object information.
27) FSPDF_PathObject_Create: used for creating an image object.
28) FSPDF_PathObject_SetPathData: used for setting image object Path data.
29) FSPDF_CanvasObject_Create: used for creating a canvas object.
30) FSPDF_CanvasObject_GetObjects: used for getting page object from canvas objects.

3.10.3 Run

Please refer to the 3.1.3 to run this demo.

3.11 Pdfobjects

3.11.1 Description

1) Purpose of demo: illustrates how to use Foxit PDF SDK to set the catalog dictionary of a PDF file.
2) Property of demo: this demo is a console application without UI.

   Usage: *pdfobjects*
3) Process: the demo loads specific PDF file from input directory. For each PDF file, the demo will modify the values of “PageLayout” and “PageMode” in catalog dictionary. Once completed, the demo will exit. All the information about execution process will be recorded into log file.
4) Input directory: used to place input files.

   Default: /samples/simple_sample/input_files. Here’re two ways:
   - i. Use default files: AboutFoxit.pdf.
   - ii. Set default sub-folder: create a sub-folder named “pdfobjects” and put all input files into this sub-folder. When running, this demo will enumerate files in sub-folder as input files.
5) Output directory: use to store output files, including log.txt.

   Default: /samples/simple_sample/output_files/pdfobjects.
6) Program files:

   pdfobjects.cpp: source codes of setting the catalog dictionary of PDF file.

3.11.2 SDK functions

The demo of pdfobjects uses following additional functions of Foxit PDF SDK:

1) FSPDF_Doc_StartSaveToFile: used for starting an incremental saving progress for PDF files.
2) FSPDF_Doc_LoadFromFile: used for loading PDF files.
3) FSPDF_Doc_GetCatalog: used for getting the catalog dictionary object in PDF files.
4) FSPDF_Dictionary_GetUnicodeString: used for getting the Unicode string value of specific attribute in PDF dictionary object.
5) FSPDF_Dictionary_SetAtUnicodeName: used for setting the Name value of specific attribute in PDF dictionary object.
6) FSPDF_Doc_Close: used for closing PDF files.

3.11.3 Run

Please refer to the 3.1.3 to run this demo.

3.12 Pdfreflow

3.12.1 Description

1) Purpose: illustrates how to use Foxit PDF SDK to reflow PDF pages.
2) Property: this demo is a console application without UI.

   It supports command line argument.

   Usage: *pdfreflow [<srcfile> /o [destfolder] /size [width,height]] /m [top,bottom,left, right] /ls [lineSpace] /s [scale] /noimage /iscontinuous]*

<srcfile>	Path of a PDF file as input file
/o [destfolder]	Path of a folder as output directory. If not set, use default output directory.
/size[width,height]	Screen size of reflowed page. If not set, default value is 480*800 pixels.
/m [top,bottom,left,right]	Margin of reflowed page. If not set, default value is [50,30,30,30] pixels
/ls [lineSpace]	Line space of reflowed page. If not set, use default line space value of SDK.
/s [scale]	Scale stage of reflowed page. The stages include 0.6, 0.8, 2, 3, 4, and 5. If not set, default value is 1.
/noimage	Set the reflowed page not to show images. If not set, images will be shown.
/iscontinuous	Set the reflowed page to display with continuous page mode. If not set, use default single page mode.

3) Process: this demo loads specified PDF files from input directory. For each PDF file, this demo will take reflow on whole pages and then save the reflowed PDF pages as bitmap files to output directory. Once completed, the demo will exit. All the information about execution process will be recorded into log file.
4) Input directory: used to place input files.
   a) Default: /samples/simple_sample/input_files. Here're two ways:
      i. Use default files: testfile.pdf.
      ii. Set default sub-folder: create a sub-folder named "pdfreflow" and put all input files into this sub-folder. When running, this demo will use files in sub-folder as input files.

   b) User-defined: set single input file through command line argument.

5) Output directory: used to store output files, including log.txt.
   a) Default: /samples/simple_sample/output_files/pdfreflow.
   b) User-defined: set path of output folder through command line argument and output directory will be "/outputfolder/pdfreflow".
6) Program files:

   pdfreflow.cpp: source code for reflowing PDF pages.

3.12.2 SDK functions

The demo of pdfreflow uses following additional functions of Foxit PDF SDK:

1) FSCRT_Bitmap_Create: used for creating a bitmap object.
2) FSCRT_Bitmap_Release: used for releasing a bitmap object.
3) FSCRT_Bitmap_FillRect: used for filling bitmap objects with specified colors.
4) FSCRT_ImageFile_Create: used for creating an image file object.
5) FSCRT_ImageFile_AddFrame: used for adding a frame image object.
6) FSCRT_ImageFile_Release: used for releasing an image file object.
7) FSCRT_PDFModule_Initialize: used for initializing a given PDF module.
8) FSCRT_PDFModule_Finalize: used for releasing a given PDF module.
9) FSPDF_Doc_StartLoad: used for loading a PDF file.
10) FSPDF_Doc_GetPage: used for getting a PDF page object.
11) FSPDF_Doc_CountPages: used for counting the number of pages of a PDF file
12) FSPDF_Doc_Close: used for closing a given PDF document.
13) FSPDF_Page_Clear: used for clearing a PDF page object but not destroying it.
14) FSPDF_Page_StartParse: used for parsing PDF page objects and returning a progress object.
15) FSPDF_ReflowPage_Create: used for creating a PDF reflow page object.
16) FSPDF_ReflowPage_SetSize: used for setting the size of screen for PDF reflowing.
17) FSPDF_ReflowPage_GetContentSize: used for getting the content size of PDF reflowed page.
18) FSPDF_RenderContext_Create: used for creating a PDF render context object.
19) FSCRT_Renderer_SetClipRect: used for setting the current clipping rectangle into a

renderer.

20) FSCRT_Renderer_CreateOnBitmap: used for creating a renderer object based on bitmap object.
21) FSPDF_ReflowPage_GetMatrix: used for getting a transformation matrix of the reflowed page.
22) FSPDF_RenderContext_SetMatrix: used for setting a transformation matrix of PDF render context object.
23) FSPDF_RenderContext_StartReflowPage: used for starting to renderer reflowed pages.
24) FSCRT_Renderer_Release: used for releasing a renderer object.
25) FSPDF_RenderContext_Release: used for releasing a PDF render context object.
26) FSPDF_ReflowPage_Release: used for releasing a PDF reflow page object.
27) FSPDF_ReflowPage_SetLineSpace: used for setting line space before calling function FSPDF_ReflowPage_StartParse.
28) FSPDF_ReflowPage_SetTopSpace: used for setting the top space of page before calling function FSPDF_ReflowPage_StartParse.
29) FSPDF_ReflowPage_StartParse: used for starting parsing process for a reflow page.

3.12.3 Run

Please refer to the 3.1.3 to run this demo.

3.12.4 Note

The reflowed pages do not support some data and text styles, such as tables, formulas, indentations, header and footer, underline and strikeout, etc.

This demo provides two modes to reflow the PDF pages: single page or continuous page. Those two modes have the same feature that if the contents cannot be displayed on one reflowed page, a new reflowed page will be created to contain the rest of the contents. The main difference among the two modes is that in continuous page, the blank area left in the current reflowed page will be filled with the contents of the next page, which can make full use of the screen space. For example, the testfile_s&screen_1_2.bmp picture is generated in single page mode, and the testfile_c&screen_2.bmp is in continuous page mode.

www.foxitsoftware.com

2

Product Overview

Foxit provides the following product offerings to our customers: **PDF Reading**

- **Foxit Reader** - Small, fast, and feature rich PDF viewer which allows you to open, view, and print any PDF file. Unlike other free PDF readers, Foxit Reader also includes easy to use collaboration features like the ability to add annotations, fill out forms, and share content with social networks. It is designed to meet the PDF viewing needs of enterprise and government organizations, making it easy to deploy and manage in large enterprise environments. Foxit Reader can also be customized to specific product features and branding.
- **Foxit MobilePDF** - Allows users to work on PDF documents while on the go with any Android or iPad device. Built on the same technology as Foxit Reader, MobilePDF provides a similar PDF user experience from desktop to mobile devices. More than just a PDF reader, it provides advanced display, collaboration tools, and page management capabilities.

Desktop PDF Editing, Creation, and Security

- **Foxit PhantomPDF™**

o **Foxit PhantomPDF *Standard*** - Expands upon PhantomPDF Express by allowing users to edit PDF document contents. Robust security features keep documents safe, including encryption and digital signature. The form design tools let users develop PDF forms. OCR support converts scanned or image-based content into selectable and searchable text. o **Foxit PhantomPDF *Business*** - Robust for the needs of businesses, it expands upon PhantomPDF Standard by providing advanced editing and security features. Policy protects PDF documents through Microsoft® Active Directory® Rights
Management Service. Permanently remove (redact) visible text and images to protect confidential information.

- **Foxit Advanced PDF Editor** - Provides paragraph editing to reflow and reformat your PDF text like a word processor. Resize page margins and spell check text to make editing PDF documents easy.

PDF Index and Search

- **Foxit PDF IFilter** - Super-fast indexing allows users to index a large amount of PDF documents and then quickly search these documents using popular search engines. Built on Microsoft®'s IFilter indexing interface. Foxit PDF IFilter extracts data from PDF documents and returns the index results to search engine software. It indexes PDF

www.foxitsoftware.com

3

testfile_s&screen_1_2.bmp **testfile_c&screen_2.bmp**

3.13 Pdfsearch

3.13.1 Description

1) Purpose: illustrates how to use Foxit PDF SDK to search specific text in the PDF files.
2) Property: this demo is a console application without UI.

   Usage: *pdfsearch*

3) Process: this demo loads specific PDF files from input directory. For each PDF file, this demo will search for the keyword "Foxit" and output search results. Once completed, the demo will exit. All the information about execution process will be recorded into log file.
4) Input directory: used to place input files.

   Default: /samples/simple_sample/input_files. Here're two ways:

   i. Use default files: AboutFoxit.pdf.
   ii. Set default sub-folder: create a sub-folder named "pdfsearch" and put all input files into this sub-folder. When running, this demo will enumerate files in sub-folder as input files.

5) Output directory: use to store output files, including log.txt.

   Default: /samples/simple_sample/output_files/pdfsearch.

6) Program files:

   pdfsearch.cpp: source codes of searching text in PDF files.

3.13.2 SDK functions

The demo of pdfsearch uses following additional functions of Foxit PDF SDK:

1) FSCRT_PDFModule_Initialize: used for initializing PDF modules.
2) FSCRT_PDFModule_Finalize: used for releasing PDF modules.
3) FSPDF_Doc_StartSaveToFile: used for starting an incremental saving progress for PDF files.
4) FSPDF_Doc_StartLoad: used for loading a PDF file.
5) FSPDF_Doc_CountPages: used for counting the number of pages of a PDF file.
6) FSPDF_Doc_GetPage: used for getting a page object of a PDF file.
7) FSPDF_Page_StartParse: used for parsing the page object of a PDF file and returning a

progress object.

8) FSPDF_TextPage_Load: used for loading text page objects of PDF files.
9) FSCRT_BStr_InitConstString: used for initializing constant string objects.
10) FSPDF_TextPage_StartSearch: used for starting searching, initializing searching environment and getting text objects of a PDF file.
11) FSPDF_TextSearch_FindNext: used for finding next matchable item.
12) FSPDF_TextSearch_GetSelection: used for getting text selection objects corresponding to the search results.
13) FSPDF_TextSelection_CountPieces: used for counting the number of pieces in the text selection objects of a PDF file.
14) FSPDF_TextSelection_GetPieceCharRange: used for getting the character range of a piece of the text selection objects.
15) FSPDF_TextSelection_GetPieceRect: used for getting the rectangular region of a piece of the text selection objects.
16) FSPDF_TextSelection_Release: used for releasing a text selection object of the PDF file.
17) FSPDF_TextSearch_Release: used for releasing a text search object of the PDF file.
18) FSPDF_TextPage_Release: used for releasing a text page object of the PDF file.
19) FSPDF_Page_Clear: used for clearing a page object of the PDF file and releasing the page resource without destroying the page object.
20) FSPDF_Doc_Close: used for closing PDF files.

3.13.3 Run

Please refer to the 3.1.3 to run this demo.

3.13.4 Note

The coordinate data achieved from the text page of the PDF file is based on the PDF page coordinate system. To highlight the search results, we need to convert the coordinate to the one based on device coordinate.

3.14 Multiple_Threads

3.14.1 Description

1) Purpose: illustrates how to use Foxit PDF SDK to open multiple PDF files in the multi-thread environment.
2) Property: this demo is a console application without UI.

   Usage: *multiple_threads*
3) Process: this demo will enumerate all PDF files in input directory. For each PDF file, this demo will create a thread to open it and load every page. Once completed, the demo will exit. All the information about execution process will be recorded into log file.
4) Input directory: used to place input files.

   Default: /samples/simple_sample/input_files.
5) Output directory: used to store output files, including log.txt.

   Default: /samples/simple_sample/output_files/multiple_threads.
6) Program files:

   multiple_threads.cpp: source code of creating threads to process PDFs in the multi-thread environment.

3.14.2 SDK functions

The demo of multiple_threads uses following additional functions of Foxit PDF SDK:

1) FSPDF_Doc_LoadFromAsyncFile: used for opening a PDF file asynchronously.
2) FSPDF_Doc_IsDocAvail: used for judging whether a document is available.
3) FSPDF_Doc_CountPages: used for counting pages of a given document.
4) FSPDF_Doc_GetPage: used for getting a page object of a PDF document.
5) FSPDF_Doc_IsPageAvail: used for judging whether a page is available.
6) FSPDF_Doc_Close: used for closing a given PDF document.
7) FSPDF_Page_StartParse: used for starting progressive control to parse pages.
8) FSPDF_Page_Clear: used for clearing a page object of a PDF document but without destroying it.

3.14.3 Run

Please refer to the 3.1.3 to run this demo.

3.14.4 Note

The processing time for each PDF may be various according to the following conditions: the running status in OS, page number and thread number etc. The max number of threads value is limited by OS.

So do not create too many threads in a process. Also, developers should choose a proper thread management.

This demo is only available for Windows, Linux and Mac.

3.15 Pdf2img

3.15.1 Description

1) Purpose: illustrates how to use Foxit PDF SDK to convert PDF files to image files that are supported by Foxit PDF SDK.
2) Property: this demo is a console application without UI.

   It supports command line argument.

   Usage: *pdf2img [<srcfile> /r [pagerange] /o [destfolder]]*

<srcfile>	Path of a PDF file as input file
/r [pagerange]	Page range, used to specify indexes of PDF pages in input PDF file, such as "0", "1-3,5". Page index starts from 0 and should be valid within page count. Invalid index will be ignored. If input page range is totally invalid, page range will be set to first page by default.
/o [destfolder]	Path of a folder as output directory. If not set, use default output directory.

3) Process: this demo loads specified PDF files from input directory. For each PDF file, this demo will convert specific PDF pages (Default: first page) to images with supported formats. Once completed, the demo will exit. All the information about execution process will be recorded into log file.
4) Input directory: used to place input files.
   a) Default: /samples/simple_sample/input_files. Here're two ways :
      i. Use default files: Pages2TIF.pdf and PDF2Img.pdf.
      ii. Set default sub-folder: create a sub-folder named "pdf2img" and put all input files into this sub-folder. When running, this demo will use files in sub-folder as input files.
   b) User-defined: set single input file through command line argument.
5) Output directory: use to store output files, including log.txt.
   a) Default: /samples/simple_sample/output_files/pdf2img.
   b) User-defined: set path of output folder through command line argument and output directory will be "/outputfolder/pdf2img".
7) Program files:

   pdf2img.cpp: source code of converting PDF to images.

3.15.2 SDK functions

The demo of pdf2img uses following additional functions of Foxit PDF SDK:

1) FSCRT_ImageFile_Create: used for creating an image file object.
2) FSCRT_ImageFile_AddFrame: used for adding a frame image.
3) FSCRT_ImageFile_Release: used for releasing an image file object.
4) FSPDF_Doc_LoadFromFile: used for loading PDF files.
5) FSPDF_Doc_CountPages: used for counting the pages of PDF documents.
6) FSPDF_Doc_GetPage: used for getting a PDF page object.
7) FSPDF_Page_Clear: used for clearing a PDF page object and releasing page resources without destroying the page object.
8) FSPDF_Doc_Close: used for closing PDF documents.
9) FSPDF_Page_GetSize: used for getting the size of pages.
10) FSCRT_Bitmap_Create: used for creating bitmap objects.
11) FSCRT_Bitmap_FillRect: used for filling bitmap objects with specified colors.
12) FSCRT_Bitmap_Release: used for releasing bitmap objects.
13) FSPDF_Page_GetMatrix: used for getting the transformation matrix of reflowed PDF pages.
14) FSPDF_Page_StartParse: used for starting to parse a PDF page object and returning a progress object.
15) FSPDF_RenderContext_Create: used for creating PDF rendering context objects.
16) FSCRT_Renderer_CreateOnBitmap: used for creating rendering objects based on bitmap objects.
17) FSPDF_RenderContext_SetMatrix: used for setting the transformation matrix of PDF rendering context objects.
18) FSPDF_RenderContext_StartPage: using PDF rendering context objects to start rendering page objects.
19) FSPDF_RenderContext_Release: used for releasing PDF rendering context objects.
20) FSCRT_PDFModule_Initialize: used for initializing PDF modules.
21) FSCRT_PDFModule_Finalize: used for releasing PDF modules.

3.15.3 Run

Please refer to the 3.1.3 to run this demo.

3.15.4 Note

Foxit PDF SDK supports converting PDF to image in five formats: BMP, JPG, TIF, PNG and JPX. The generated TIF files can include data of one-frame or multi-frame.

3.16 Img2pdf

3.16.1 Description

1) Purpose : illustrates how to use Foxit PDF SDK to convert image files to PDF files.
2) Property: this demo is a console application without UI.

   It supports command line argument.

   Usage: *img2pdf [<srcfile> /o [destfolder]]*

   <srcfile> Path of an image file as input file.

   /o [destfolder] Path of a folder as output directory. If not set, use default output directory.
3) Process: the demo loads specific image files from input directory. Then each image will be converted to a PDF and stored into output directory. Once completed, the demo will exit. All the information about execution process will be recorded into log file.
4) Input directory: used to place input files.
   a) Default: /samples/simple_sample/input_files. Here’re two ways:
      i. Use default files: 1.jpg, 2.jpg, 3.jpg and TIF2Pages.tif.
      ii. Set default sub-folder: create a sub-folder named “img2pdf” and put all input files into this sub-folder. When running, this demo will enumerate files in sub-folder as input files.
   b) User-defined: set single input file through command line argument.
5) Output directory: used to store output files, including log.txt.
   a) Default: /samples/simple_sample/output_files/img2pdf.
   b) User-defined: set path of output folder through command line argument and output directory will be “/outputfolder/img2pdf”.
6) Program files:

   img2pdf.cpp: source code of converting the images to PDFs.

3.16.2 SDK functions

The demo of img2pdf uses following additional functions of Foxit PDF SDK:

1) FSCRT_ImageFile_Create: used for creating an image file object.
2) FSCRT_ImageFile_AddFrame: used for adding a frame image.

3) FSCRT_ImageFile_Release: used for releasing an image file object.
4) FSCRT_Image_LoadFromFile: used for loading images from image files.
5) FSCRT_Image_CountFrames: used for counting the number of frames in the specified image.
6) FSCRT_Image_LoadFrame: used for loading a frame of data in the specified image.
7) FSCRT_PDFModule_Initialize: used for initializing PDF modules.
8) FSCRT_PDFModule_Finalize: used for releasing PDF modules.
9) FSPDF_Doc_StartSaveToFile: used for starting an incremental saving progress for PDF files.
10) FSPDF_Page_Clear: used for clearing a page object of a PDF file and releasing page resources, but without destroying the page object.
11) FSPDF_Doc_Create: used for creating a new PDF document object.
12) FSPDF_Doc_Close: used for closing PDF documents.
13) FSPDF_Page_StartParse: used for starting to parse a PDF page object and returning a progress object.
14) FSPDF_ImageObject_Create: used for creating an image object.
15) FSPDF_ImageObject_SetImage: used for associating the image data with the image object.
16) FSCRT_Image_GetCurrentFrameSize: used for getting the size of the current frame in the image.
17) FSPDF_PageObject_SetMatrix: used for setting transformation matrix of page objects.
18) FSPDF_Page_GetPageObjects: used for getting the page objects in the specified pages.
19) FSPDF_PageObjects_InsertObject: used for inserting an object to a specified page object.
20) FSPDF_PageObjects_GenerateContents: used for generating PDF page contents.
21) FSPDF_Page_Create: used for creating a new page.
22) FSPDF_Page_SetSize: used for setting the page size.

3.16.3 Run

Please refer to the 3.1.3 to run this demo.

3.16.4 Note

Foxit PDF SDK supports converting following six types of images to PDF: BMP, JPG, GIF, TIF, PNG and JPX. If the TIF image contains multiple frames, it will generate a page for each frame in PDF.

3.17 Pdfencrypt

3.17.1 Description

1) Purpose: illustrates how to use Foxit PDF SDK to set password encryption, custom encryption, Foxit DRM encryption, certification encryption and user permission encryption for a PDF file. In addition, decryption of PDF is also included.
2) Property: this demo is a console application without UI.

   It supports command line argument.

   Usage: *pdfencryp [<srcfile> /o [destfolder]]*

<srcfile>	Path of a PDF file as input file.
/o [destfolder]	Path of a folder as output directory. If not set, use default output directory.

3) Process: the demo loads specified PDF files from input directory. For each PDF file, this demo will encrypt the PDF respectively by password, custom encryption solution, Foxit DRM, security certificate or permission and then save the encrypted PDFs to output directory. In addition, it will decrypt the encypted PDFs respectively and then save the decrypted PDFs as well. Once completed, the demo will exit. All the information about execution process will be recorded into log file.
4) Input directory: used to place input files.
   a) Default: /samples/simple_sample/input_files. Here're two ways:
      i. Use default files: AboutFoxit.pdf.
      ii. Set default sub-folder: create a sub-folder named "pdfencypt" and put all input files into this sub-folder. When running, this demo will enumerate files in sub-folder as input files
   b) User-defined: set single input file through command line argument.
5) Output directory: used to store output files, including log.txt.
   a) Default: /samples/simple_sample/output_files/pdfencypt.
   b) User-defined: set path of output folder through command line argument and output directory will be "/outputfolder/pdfencypt".
6) Program files:

   pdfencrypt.cpp: source code of encrypting PDF by password, custom encryption, Foxit DRM, security certificate or permission, and decrypting PDF.

3.17.2 SDK functions

The demo of pdfencrypt uses following additional functions of Foxit PDF SDK:

1) FSCRT_PDFModule_Initialize: used for initializing PDF module.
2) FSCRT_PDFModule_Finalize: used for finalizing PDF module.
3) FSPDF_Doc_StartSaveToFile: used for starting an incremental saving progress for PDF files.
4) FSPDF_Doc_LoadFromFile: used for loading PDF files.
5) FSPDF_Doc_Close: used for closing PDF files.
6) FSPDF_Security_StartPasswordEncryption: used for encrypting and setting user rights for PDF files.
7) FSPDF_Security_StartCustomEncryption: used for encrypting the PDF document by certificate.
8) FSPDF_Security_StartCertificateEncryption: used for encrypting the PDF document by certificate.
9) FSPDF_Security_StartFoxitDRMEncryption: used for starting PDF document encryption by using Foxit DRM.
10) FSPDF_Security_RegisterHandler: used for registering a customer security handler to Foxit PDF SDK, enabling access to a PDF document which is protected by customized security handler.
11) FSPDF_Security_UnregisterHandler: used for unregistering a customer security handler to Foxit PDF SDK.
12) FSPDF_Security_SetFoxitDRMHandler: used for setting Foxit DRM security handler to Foxit PDF SDK.
13) FSPDF_Security_SetCertificateHandler: used for setting certificate security handler to Foxit PDF SDK.

3.17.3 Run

Please refer to the 3.1.3 to run this demo.

3.17.4 Note

- On Windows, PDF file encrypted by security certificate can be opened directly if foxit_all.pfx is installed, which can be found in: /samples/simple_sample/input_files.
- Before running this demo, please make sure that you have already installed OpenSSL.

  For MAC OS X, please install OpenSSL with 64-bit, not 32-bit.

  On MAC OS X and Linux, you can also download an OpenSSL source package, for example:

openssl-0.9.8.tar.gz, and then configure prefix path to /usr/local. The default openssl path is /usr/local/ssl in Makefile file, if you change the openssl installation path, please update Makefile to new path when executing "make pdfencrypt" command.

On iOS, you should put openssl package into the directory “lib” of foxit pdf sdk package. The default paths of Xcode configuration of the demo are "$(SRCROOT)/../../../lib/ssl/lib" and "$(SRCROOT)/../../../lib/ssl/include”. If you change the openssl package path, please update the Xcode configuration.

3.18 Pdfprint

3.18.1 Description

1) Purpose: illustrates how to use Foxit PDF SDK to open and print PDF files. Only available for Windows.
2) Property: this demo is a console application without UI.

   It supports command line argument.

   Usage: *pdfprint [<srcfile> /r [pagerange] /o [destfolder]]*

<srcfile>	Path of a PDF file as input file
/r [pagerange]	Page range, used to specify indexes of PDF pages in input PDF file, such as "0", "1-3,5". Page index starts from 0 and should be valid within page count. Invalid index will be ignored. If input page range is totally invalid, page range is to be first page by default.
/o [destfolder]	Path of a folder as output directory. If not set, use default output directory.

3) Process: this demo loads PDF files from the input directory. For each PDF file, this demo will print specific pages (Default: first page). Once completed, the demo will exit. All the information about execution process will be recorded into log file.
4) Input directory: used to place input files.
   a) Default: /samples/simple_sample/input_files. Here're two way:
      i. Use default files: AboutFoxit.pdf.
      ii. Set default sub-folder: create a sub-folder named "pdfprint" and put all input files into this sub-folder. When running, this demo will enumerate files in sub-folder as input files.
   b) User-defined: set single input file through command line argument
5) Output directory: used to store output files, including log.txt.
   a) Default: /samples/simple_sample/output_files/pdfprint.
   b) User-defined: set path of output folder through command line argument and output directory will be "/outputfolder/pdfprint".
6) Program files:

   pdfprint.cpp: source codes of opening and printing PDF files.

3.18.2 SDK functions

The demo of pdfprint uses following additional functions of Foxit PDF SDK:

1) FSPDF_Doc_StartSaveToFile: used for starting an incremental saving progress for PDF files.
2) FSPDF_Doc_StartLoad: used for loading a PDF file.
3) FSPDF_Doc_GetPage: used for getting a PDF page object.
4) FSPDF_Page_StartParse: used for parsing a PDF page object and returning a progress object.
5) FSPDF_RenderContext_Create: used for creating a PDF render context object.
6) FSCRT_Renderer_CreateOnWindowsDC: used for creating a render object based on windows device context.
7) FSPDF_Page_GetMatrix: used for getting a transformation matrix of PDF page.
8) FSPDF_RenderContext_SetMatrix: used for setting a transformation matrix of PDF render context object.
9) FSPDF_RenderContext_StartPage: used for starting a step-by-step progress for PDF page rendering.
10) FSCRT_Renderer_Release: used for releasing a render object.
11) FSPDF_RenderContext_Release: used for releasing a PDF render context object.
12) FSPDF_Page_Clear: used for clearing a PDF page object, and releasing page resources without destroying the page object.
13) FSPDF_Doc_Close: used for closing a PDF file.

3.18.3 Run

Please refer to the 3.1.3 to run this demo.

3.18.4 Note

This demo is only available for Windows and will use the default system printer.

3.19 Pdfwatermark

3.19.1 Description

1) Purpose: illustrates how to use Foxit PDF SDK to add different types of watermarks into PDF files.

2) Property: this demo is a console application without UI.

   Usage: *pdfwatermark*

3) Process: this demo loads specific PDF files from input directory. For each PDF file, this demo will add three types of watermarks which are text, image and page watermarks, and then generate a new PDF with newly adding watermark for each type. Once completed, the demo will exit. All the information about execution process will be recorded into log file.

4) Input directory: used to place input files. Specially, please ensure that FoxitLog.jpg and FoxitBigpreview.pdf are always available in input directory.

   Default: /samples/simple_sample/input_files. Here're two ways:

   i. Use default files: AboutFoxit.pdf

   ii. Set default sub-folder: create a sub-folder named "pdfwatermark" and put all input files into this sub-folder. When running, this demo will use files in sub-folder as input files.

5) Output directory: used to store output files, including log.txt.

   Default: /samples/simple_sample/output_files/pdfwatermark.

6) Program files:

   pdfwatermark.cpp: source codes of adding different types of watermarks and saving PDF files with newly added watermarks.

3.19.2 SDK functions

The demo of pdfwatermark uses following additional functions of Foxit PDF SDK:

1) FSCRT_PDFModule_Initialize: used for initializing a PDF module.
2) FSCRT_PDFModule_Finalize: used for releasing a PDF module.
3) FSCRT_Font_CreateStandard: used for creating a standard font object.
4) FSCRT_Font_Release: used for releasing a font object.
5) FSPDF_Doc_StartSaveToFile: used for starting an incremental saving progress for PDF files.

6) FSPDF_Doc_LoadFromFile: used for loading a PDF file.
7) FSPDF_Doc_CountPages: used for counting the number of pages of a PDF file.
8) FSPDF_Doc_GetPage: used for getting a page object of a PDF file.
9) FSPDF_Page_Clear: used for clearing a page object of PDF file and releasing the page resource without destroying the page object.
10) FSPDF_Page_StartParse: used for parsing a given page.
11) FSPDF_Doc_Close: used for closing a given PDF file.
12) FSPDF_Watermark_CreateFromText: used for creating a Text Watermark object.
13) FSPDF_Watermark_InsertToPage: used for inserting watermarks into a given page.
14) FSPDF_Watermark_CreateFromImage: used for creating an Image Watermark object.
15) FSPDF_Watermark_CreateFromPage: used for creating Page Watermark objects.
16) FSPDF_Watermark_Release: used for releasing watermark objects.
17) FSCRT_Image_LoadFromFile: used for loading image files and creating image objects.
18) FSCRT_Image_Release: used for releasing image objects.

3.19.3 Run

Please refer to the 3.1.3 to run this demo.

3.20 Pdfasync

3.20.1 Description

1) Purpose: illustrates how to use Foxit PDF SDK to open local PDF documents asynchronously.
2) Property: this demo is a console application without UI.

   Usage: *pdfasync*
3) Process: this demo will load specific PDF files from input directory asynchronously and then parse pages one by one. Once completed, the demo will exit. All the information about execution process will be recorded into log file.
4) Input directory: used to place input files.

   Default: /samples/simple_sample/input_files. Here're two ways:

   i. Use default files: FoxitBigPreview.pdf.
   ii. Set default sub-folder: create a sub-folder named "pdfasync" and put all input files into this sub-folder. When running, this demo will enumerate files in sub-folder as input files.
5) Output directory: used to store output files, including log.txt.

   Default: /samples/simple_sample/output_files/pdfasync.
6) Program file:

   pdfasync.cpp: source code of opening PDF files asynchronously and parsing pages one by one.

3.20.2 SDK functions

The demo of pdfasync uses following additional functions of Foxit PDF SDK:

1) FSPDF_Doc_LoadFromAsyncFile: used for opening a PDF file asynchronously.
2) FSPDF_Doc_IsDocAvail: used for judging whether a document is available.
3) FSPDF_Doc_CountPages: used for counting pages of a given document.
4) FSPDF_Doc_GetPage: used for getting a page object of a PDF document.
5) FSPDF_Doc_IsPageAvail: used for judging whether a page is available.
6) FSPDF_Doc_Close: used for closing a given PDF document.
7) FSPDF_Page_StartParse: used for starting progressive control to parse pages.
8) FSPDF_Page_Clear: used for clearing a page object of a PDF document but without destroying it.

3.20.3 Run

Please refer to the 3.1.3 to run this demo.

3.21 Fdf

3.21.1 Description

1) Purpose: illustrates how to use Foxit PDF SDK to import annotations from FDF files and export annotations to FDF files.
2) Property: this demo is a console application without UI.

   Usage: *fdf*
3) Process: this demo loads specific PDF and FDF files from input directory. For each FDF file, this demo will open the PDF file which is related with FDF file and import annotations from FDF file to this PDF file. For each PDF file, this demo will create a FDF file and export the annotations from PDF file to the newly created FDF file. Once completed, the demo will exit. All the information about execution process will be recorded into log file.
4) Input directory: used to place input files.

   Default: /samples/simple_sample/input_files. Here're two ways:

   i. Use default files:

      AnnotationData.fdf, InsertPages.pdf , AnnotationDataExport.pdf.

   ii. Set default sub-folder: create a sub-folder named "fdf" and put all input files into this sub-folder. When running, this demo will use files in sub-folder as input files. Specially, please ensure the PDF files which are related with FDF files are placed in the right place.
5) Output directory: used to store output files, including log.txt.

   Default: /samples/simple_sample/output_files/fdf.
6) Program file:

   fdf.cpp: source code of importing PDF annotations from FDF files and exporting PDF annotations to FDF files.

3.21.2 SDK functions

The demo of fdf uses following additional functions of Foxit PDF SDK:

1) FSCRT_ImageFile_Create: used for creating an image file object.
2) FSCRT_ImageFile_AddFrame: used for adding a frame image.
3) FSCRT_ImageFile_Release: used for releasing an image file object.
4) FSPDF_Doc_StartSaveToFile: used for starting an incremental saving progress for PDF files.

5) FSPDF_Doc_LoadFromFile: used for loading PDF documents.
6) FSPDF_Doc_CountPages: used for counting the number of pages in a given PDF document.
7) FSPDF_Doc_GetPage: used for getting a page object of a given PDF document.
8) FSPDF_Page_Clear: used for clearing a page object of a PDF document and releasing page resources, but without destroying page object.
9) FSPDF_Doc_Close: used for closing a PDF document.
10) FSFDF_Doc_Load: used for loading a FDF file.
11) FSFDF_Doc_Create: used for creating a FDF file.
12) FSFDF_Doc_GetPDFPath: used for getting a corresponding PDF path of a given FDF file.
13) FSPDF_Doc_CountAnnots: used for counting annotations in a given document.
14) FSPDF_Doc_GetAnnot: used for getting the specified annotations.
15) FSFDF_Annot_ExportToPDFPage: used for importing the specified annotations to a PDF pages from FDF file.
16) FSFDF_Doc_Close: used for closing a FDF file.
17) FSFDF_Doc_Save: used for saving a FDF file.
18) FSPDF_Page_LoadAnnots: used for loading annotations in a specified page.
19) FSPDF_Page_UnloadAnnots: used for releasing annotations in a specified page.
20) FSPDF_Annot_ImportToPDFPage: used for importing annotations to a specified page.
21) FSPDF_Annot_GetByFilter: used for getting annotations by specifying Filter and sequence number.
22) FSPDF_Annot_ExportToFDFDoc: used for exporting annotations to FDF files.

3.21.3 Run

Please refer to the 3.1.3 to run this demo.

3.21.4 Note

FDF files save the path of PDF files, annotations from which is exported to the FDF files. So before running this demo, please ensure that the original PDF files are in the right place so that annotations can be exported from FDF file to PDF file correctly.

3.22 Psi

3.22.1 Description

1) Purpose: illustrates how to use Foxit PDF SDK to add pressure sensitive ink.
2) Property: this demo is a console application without UI.

   Usage: *psi*
3) Process: this demo loads specific PDF files from input directory. For each PDF file, this demo will add a PSI object into the first page and then save it as a new PDF file to output directory. Once completed, the demo will exit. All the information about execution process will be recorded into log file.
4) Input directory: used to place input files.

   Default: /samples/simple_sample/input_files. Here're two ways:

   i. Use default files: AboutFoxit.pdf.
   ii. Set default sub-folder: create a sub-folder named "psi" and put all input files into this sub-folder. When running, this demo will enumerate files in sub-folder as input files.
5) Output directory: used to store output files, including log.txt.

   Default: /samples/simple_sample/output_files/psi.
6) Program files:

   psi.cpp: source code of adding pressure sensitive ink.

3.22.2 SDK functions

The demo of psi uses following additional functions of Foxit PDF SDK:

1) FSCRT_PDFModule_Initialize: used for initializing a PDF module.
2) FSCRT_PDFModule_Finalize: used for releasing a PDF module.
3) FSPDF_Doc_StartSaveToFile: used for starting an incremental saving progress for PDF files.
4) FSPDF_Doc_StartLoad: used for loading a PDF file.
5) FSPDF_Doc_GetPage: used for getting a page object of a PDF file.
6) FSPDF_Page_Clear: used for releasing a page object without destroying the page object.

7) FSPDF_Page_GetMatrix: used for getting the transformation matrix of a page.
8) FSPDF_Doc_Close: used for closing a given PDF file.
9) FSCRT_PSI_Create: used for creating a pressure sensitive ink object.
10) FSCRT_PSI_InitCanvas: used for initializing the canvas size of pressure sensitive ink.
11) FSCRT_PSI_SetInkColor: used for setting color of the pressure sensitive ink.
12) FSCRT_PSI_SetInkDiameter: used for setting the diameter of pen points of pressure sensitive ink.
13) FSCRT_PSI_SetOpacity: used for setting the opacity of pressure sensitive ink.
14) FSCRT_PSI_AddPoint: used for adding a point of pressure sensitive ink.
15) FSCRT_PSI_GetContentsRect: used for getting the size of the rectangular region of pressure sensitive ink in a device.
16) FSCRT_PSI_Release: used for releasing a pressure sensitive ink object.
17) FSCRT_Matrix_GetReverse: used for getting an inverse matrix from a given matrix.
18) FSCRT_Matrix_TransformRectF: used for transforming the matrix of a given rectangle.
19) FSCRT_PSI_ConvertToPDFAnnot: used for converting pressure sensitive ink to an annotation.

3.22.3 Run

Please refer to the 3.1.3 to run this demo.

3.23 Pdfflatten

3.23.1 Description

1) Purpose: illustrates how to use Foxit PDF SDK to flatten a PDF page, which means to make annotations and form fields to be a part of page contents.
2) Property: this demo is a console application without UI.

   It supports command line argument.

   Usage: *pdfflatten [<srcfile> /r [pagerange] /o [destfolder]]*

<srcfile>	Path of a PDF file as input file
/r [pagerange]	Page range, used to specify indexes of PDF pages in input PDF file, such as "0", "1-3,5". Page index starts from 0 and should be valid within page count. Invalid index will be ignored. If input page range is totally invalid, page range is to be first page by default.
/o [destfolder]	Path of a folder as output directory. If not set, use default output directory.

3) Process: this demo loads specific PDF files from input directory. For each PDF file, this demo will flatten annotation and form objects in specific pages (Default: first page) and then save the changed PDF as a new one to output directory. Once completed, the demo will exit. All the information about execution process will be recorded into log file.
4) Input directory: used to place input files.
   a) Default: /samples/simple_sample/input_files. Here're two ways :
      i. Use default files: Annot_Mixed.pdf.
      ii. Set default sub-folder: create a sub-folder named “pdfflatten” and put all input files into this sub-folder. When running, this demo will enumerate files in sub-folder as input files.
   b) User-defined: set single input file through command line argument.
5) Output directory: used to store output files, including log.txt.
   a) Default: /samples/simple_sample/output_files/pdfflatten.
   b) User-defined: set path of output folder through command line argument and output directory will be “/outputfolder/pdfflatten”.
6) Program files

   pdfflatten.cpp: source code of flattening annotation and form objects.

3.23.2 SDK functions

The demo of pdfflatten uses following additional functions of Foxit PDF SDK:

1) FSPDF_Doc_StartSaveToFile: used for starting an incremental saving progress for PDF files.
2) FSPDF_Doc_StartLoad: used for loading a PDF file.
3) FSPDF_Doc_CountPages: used for getting the page count of a document.
4) FSPDF_Doc_GetPage: used for getting a page object.
5) FSPDF_Page_Clear: used for clearing a page object of a PDF file and releasing page resources but without destroying the page object.
6) FSPDF_Doc_Close: used for closing a PDF document.
7) FSPDF_Page_Flatten: used for flattening a PDF page and annotations or form fields.
8) FSCRT_PDFModule_Initialize: used for initializing a PDF module.
9) FSCRT_PDFModule_Finalize: used for releasing a PDF module.

3.23.3 Run

Please refer to the 3.1.3 to run this demo.

3.24 Pdfwrapper

3.24.1 Description

1) Purpose: illustrates how to use Foxit PDF SDK to create a wrapper PDF document and open a wrapper document.
2) Property: this demo is a console application without UI.

   Usage: *pdfwrapper*
3) Process: this demo loads specific PDF files from input directory. For each PDF file, this demo will make a wrapper for it and save the generated wrapper file to output directory. Then this demo will load the wrapper PDF file and render for first page of wrapper part and first page of real PDF file. Once completed, the demo will exit. All the information about execution process will be recorded into log file.
4) Input directory: use to place input files. Specially, please ensure wrapper.pdf is always available in input directory.

   Default: /samples/simple_sample/input_files. Here're two ways:

   i. Use default files: AboutFoxit.pdf.
   ii. Set default sub-folder: create a sub-folder named "pdfwrapper" and put all input files into this sub-folder. When running, this demo will enumerate files in sub-folder as input files.
5) Output directory: use to store output files, including log.txt.

   Default: /samples/simple_sample/output_files/pdfwrapper.
6) Program files:

   pdfwrapper.cpp: source code of creating wrapper PDF file and loading it.

3.24.2 SDK functions

The demo of pdfwrapper uses following additional functions of Foxit PDF SDK:

1) FSCRT_ImageFile_Create: used for creating an image file object.
2) FSCRT_ImageFile_AddFrame: used for adding a frame image.
3) FSCRT_ImageFile_Release: used for releasing an image file object.
4) FSPDF_Doc_StartSaveToFile: used for starting an incremental saving progress for PDF files.
5) FSPDF_Doc_LoadFromFile: used for loading a PDF file.
6) FSPDF_Doc_CountPages: used for counting the pages of PDF documents.

7) FSPDF_Doc_GetPage: used for getting a PDF page object.
8) FSPDF_Page_Clear: used for clearing a PDF page object and releasing page resources without destroying a page object.
9) FSPDF_Doc_Close: used for closing a given PDF document.
10) FSPDF_Page_GetSize: used for getting the size of pages.
11) FSCRT_Bitmap_Create: used for creating bitmap objects.
12) FSCRT_Bitmap_FillRect: used for filling bitmap objects with the specified colors.
13) FSCRT_Bitmap_Release: used for releasing bitmap objects.
14) FSPDF_Page_GetMatrix: used for getting the transformation matrix of PDF reflowed pages.
15) FSPDF_Page_StartParse: used for starting to parse a PDF page object and returning a progress object.
16) FSPDF_RenderContext_Create: used for creating PDF rendering context objects.
17) FSCRT_Renderer_CreateOnBitmap: used for creating rendering objects based on bitmap objects.
18) FSPDF_RenderContext_SetMatrix: used for setting the transformation matrix of PDF rendering context objects.
19) FSPDF_RenderContext_StartPage: using PDF rendering context objects to start rendering page objects.
20) FSPDF_RenderContext_Release: used for releasing PDF rendering context objects.
21) FSPDF_Doc_SaveAsWrapperFile: used for saving a PDF document as a wrapper file.
22) FSPDF_Doc_GetWrapperOffset: used for getting wrapper offset.
23) FSPDF_Doc_IsWrapper: used for determining whether a document is a wrapper document or not.
24) FSPDF_Doc_GetWrapperData： used for getting the wrapper data from a given document.
25) FSCRT_PDFModule_Initialize: used for initializing a PDF module.
26) FSCRT_PDFModule_Finalize: used for releasing a PDF module.

3.24.3 Run

Please refer to the 3.1.3 to run this demo.

3.24.4 Note

Foxit PDF SDK supports inserting a PDF file to another one as a wrapper PDF file. There is no way to load this kind of PDF file directly, only the wrapper file can be loaded. By using wrapper functions of Foxit PDF SDK, the real content of PDF file can be loaded.

3.25 Pdfattachments

3.25.1 Description

1) Purpose: illustrates how to use Foxit PDF SDK to save PDF's attachments as new files and add a file into PDF file as an attachment.
2) Property: this demo is a console application without UI.

   Usage: *pdfattachments*
3) Process: this demo loads specific PDF files from input directory. For each PDF file, this demo will access its attachments and save them as new files. In addition, it will add a specified file to the opened PDF as an attachment and then save the changed PDF as a new PDF file to output directory. Once completed, the demo will exit. All the information about execution process will be recorded into log file.
4) Input directory: used to place input files. Specially, please ensure that FoxitLogo.jpg is always available in input directory.

   Default: /samples/simple_sample/input_files. Here're two ways:

   i. Use default files: AttachMent.pdf.

   ii. Set default sub-folder: create a sub-folder named "pdfattachments" and put all input files into this sub-folder. When running, this demo will use files in sub-folder as input files.
5) Output directory: used to store output files, including log.txt.

   Default: /samples/simple_sample/output_files/pdfattachments.
6) Program file：

   pdfattachments.cpp: source code of saving PDF's attachments as new files and adding a file into PDF as an attachment.

3.25.2 SDK functions

The demo of pdfattachments uses following additional functions of Foxit PDF SDK:

1) FSPDF_Doc_StartSaveToFile: used for starting an incremental saving progress for PDF files.
2) FSPDF_Doc_StartLoad: used for loading a PDF file.
3) FSPDF_Doc_LoadAttachments: used for loading attachments from a given PDF file.
4) FSPDF_Attachments_CountAttachment : used for counting attachments.
5) FSPDF_Attachments_GetAttachment: used for getting the attachment according to the given index.

6) FSPDF_Attachments_Release: used for releasing attachments.
7) FSPDF_Attachment_GetFileName: used for getting an attachment's name.
8) FSCRT_UTF8_DecodeToUTF16LE: used for decoding UTF8 to UTF16LE.
9) FSPDF_Attachment_WriteToFile: used for saving attachments.
10) FSPDF_Attachments_InsertAttachment: used for inserting an attachment to a PDF file.
11) FSPDF_Attachment_SetFile: used for setting a file to an attachment.
12) FSPDF_Attachment_SetFileName: used for setting the file name of a given attachment.
13) FSPDF_Doc_Close: used for closing a PDF file.
14) FSCRT_PDFModule_Initialize: used for initializing a PDF module.
15) FSCRT_PDFModule_Finalize: used for releasing a PDF module.

3.25.3 Run

Please refer to the 3.1.3 to run this demo.

3.26 Pdfcontentmark

3.26.1 Description

1) Purpose: illustrates how to use Foxit PDF SDK to mark PDF content.
2) Property: this demo is a console application without UI.

   Usage: *pdfcontentmark*
3) Process: the demo creates a new PDF and adds a text object to the PDF page. Then it will mark the text object and save the PDF to output directory. Once completed, the demo will exit. All the information about execution process will be recorded into log file.
4) Input directory: no input file is needed.
5) Output directory: used to store output files, including log.txt.

   Default: /samples/simple_sample/output_files/pdfcontentmark.
6) Program files:

   pdfcontentmark.cpp: source code of marking PDF contents.

3.26.2 SDK functions

The demo of pdfcontentmark uses following additional functions of Foxit PDF SDK:

1) FSPDF_Doc_StartSaveToFile: used for starting an incremental saving progress for PDF files.
2) FSPDF_Page_Clear: used for clearing a PDF page object and releasing page resources without destroying the page object.
3) FSPDF_Doc_Close: used for closing PDF documents.
4) FSPDF_Doc_Create: used for creating PDF documents.
5) FSPDF_Page_Create: used for creating PDF pages.
6) FSPDF_Page_SetSize: used for setting page’s width and height.
7) FSPDF_Page_GetPageObjects: used for getting page objects in a PDF page.
8) FSPDF_TextObject_Create: used for creating a new text object.
9) FSCRT_Font_Create: used for creating a font with attributes.
10) FSPDF_TextObject_SetTextState: used for setting text states of a text object.
11) FSPDF_TextObject_SetUnicodeString: used for setting Unicode string for a text object.
12) FSPDF_PageObject_SetMatrix: used for setting matrix of a page object.
13) FSCRT_Font_Release: used for releasing a font object.

14) FSPDF_PageObjects_InsertObject: used for inserting a page object and it will be automatically freed if success.
15) FSPDF_PageObjects_GenerateContents: used for generating PDF page content.
16) FSPDF_PageObject_GetMarkedContent: used for getting the marked content object from a page object.
17) FSPDF_MarkedContent_CountItems: used for getting the count of marked content items in the marked content sequence.
18) FSPDF_Object_CreateDict: used for creating a dictionary PDF object.
19) FSPDF_Dictionary_SetAtInteger: used for setting an integer value to specific key in dictionary.
20) FSPDF_MarkedContent_AddItem: used for adding a new marked content item to the current marked content object.
21) FSPDF_MarkedContent_GetTagName: used for getting the name of a specific marked content item.
22) FSPDF_MarkedContent_GetItemParam: used for getting the parameter type of the value of a specific marked content item.
23) FSCRT_PDFModule_Initialize: used for initializing PDF modules.
24) FSCRT_PDFModule_Finalize: used for releasing PDF modules.

3.26.3 Run

Please refer to the 3.1.3 to run this demo.

3.27 Pdflayer

3.27.1 Description

1) Purpose: illustrates how to use Foxit PDF SDK to render all PDF layers of a PDF document.
2) Property: this demo is a console application without UI.

   Usage: *pdflayer*
3) Process: this demo loads specific PDF file from input directory. For the PDF file, this demo will create a view layer context and enumerate all PDF layers. Then it will traverse layer tree and set the opposite visible state of every PDF layer. Finally, render each page to image and save the image to output directory. Once completed, the demo will exit. All the information about execution process will be recorded into log file.
4) Input directory: used to place input files.

   Default: /samples/simple_sample/input_files. Here're two ways:
   i. Use default files: PDFLayer_3c.pdf.
   ii. Set default sub-folder: create a sub-folder named "pdflayer" and put all input files into this sub-folder. When running, this demo will enumerate files in sub-folder as input files.
5) Output directory: used to store output files, including log.txt.

   Default: /samples/simple_sample/output_files/pdflayer.
6) Program files:

   pdflayer.cpp: source code of rending all PDF layers of a PDF document.

3.27.2 SDK functions

The demo of pdflayer uses following additional functions of Foxit PDF SDK:

1) FSCRT_Bitmap_Create: used for creating a bitmap.
2) FSCRT_Bitmap_FillRect: used for filling a bitmap with a specified color.
3) FSCRT_Bitmap_Release: used for destroying a bitmap and releasing related resource.
4) FSPDF_Doc_CountPages: used for getting the page count of a document.
5) FSPDF_Doc_GetPage: used for getting a page object.
6) FSPDF_Doc_StartLoad: used for loading a PDF file.
7) FSPDF_Doc_EnumLayers: used for enumerating all PDF layers.
8) FSPDF_Doc_Close: used for closing a PDF document.

9) FSCRT_Renderer_CreateOnBitmap: used for creating a renderer on a bitmap object.
10) FSCRT_Renderer_Release: used for releasing a given renderer object.
11) FSPDF_RenderContext_SetMatrix: used for setting a transformation matrix to a rendering context.
12) FSPDF_RenderContext_SetLayerContext: used for setting layer context to render context handle.
13) FSPDF_RenderContext_Create: used for creating a PDF rendering context.
14) FSPDF_RenderContext_Release: used for releasing a PDF rendering context.
15) FSPDF_RenderContext_StartPage: used for starting rendering a PDF page in a renderer with a PDF rendering context.
16) FSPDF_Page_GetMatrix: used for getting page transformation matrix.
17) FSPDF_Page_StartParse: used for starting parsing a PDF page.
18) FSPDF_Page_GetSize: used for getting page size.
19) FSPDF_Page_Clear: used for releasing all page contents and relative resources.
20) FSPDF_LayerContext_Create: used for creating a PDF layer context with a given type.
21) FSPDF_LayerNode_Init: used for initializing a PDF layer node.
22) FSPDF_LayerNode_Clear: used for clearing a PDF layer node.
23) FSPDF_Layer_GetName: used for getting a name of a PDF layer.
24) FSPDF_LayerContext_IsVisible: used for checking whether a PDF layer is visible or not.
25) FSPDF_LayerContext_SetVisible: used for changing a PDF layer visibility state.
26) FSPDF_LayerContext_Release: used for releasing a PDF layer context.

3.27.3 Run

Please refer to the 3.1.3 to run this demo.

3.28 Matrix

3.28.1 Description

1) Purpose: illustrates how to transform page objects, like image and text objects, in PDF documents using Matrix.
2) Property: this demo is a console application without UI.

   Usage: *matrix*
3) Process: this demo loads specific PDF file from input directory. For the PDF file, this demo will search for the image object of "Foxit" logo and text object of "Foxit Software Incorporated". Then the objects will be transformed by four matrices respectively. The matrices are translation, rotation, scale and skew. Finally, the transformed results will be rendered to the source PDF document which will be saved to the output directory. Once completed, the demo will exit.
4) Input directory: used to place input files.

   Default: /samples/simple_sample/input_files.

   Use default files: SamplePDF.pdf.
5) Output directory: used to store output files.

   Default: /samples/simple_sample/output_files/matrix.

   Default output files: MatrixTransformResult.pdf.
6) Program files:

   matrix.cpp: source code of transforming page objects.

3.28.2 SDK functions

The demo of matrix uses following additional functions of Foxit PDF SDK:

1) FSPDF_Doc_StartLoad: used for loading a PDF file.
2) FSPDF_Doc_GetPage: used for getting a page object.
3) FSPDF_Page_Clear: used for releasing all page contents and relative resources.
4) FSPDF_Page_StartParse: used for starting parsing a PDF page.
5) FSPDF_Page_GetPageObjects: used for getting page objects in a PDF page.
6) FSPDF_PageObjects_CountObjects: used for getting the count of page objects with specific type.
7) FSPDF_PageObjects_GetObject: used for getting a page object from page objects
8) FSPDF_PageObject_GetType: used for getting type of a page object.

9) FSPDF_TextObject_GetUnicodeString: used for getting Unicode string of a text object.
10) FSPDF_PageObject_Clone: used for cloning a new page object.
11) FSPDF_PageObject_GetMatrix: used for getting matrix of a page object.
12) FSPDF_PageObject_SetMatrix: used for setting matrix of a page object.
13) FSPDF_PageObjects_InsertObject: used for inserting a page object and it will be automatically freed.
14) FSPDF_PageObject_GetRect: used for getting rectangle of a page object.
15) FSPDF_PageObjects_GenerateContents: used for generating PDF Page content.
16) FSCRT_Matrix_Translate: used for translating a coordinate matrix.
17) FSCRT_Matrix_Rotate: used for rotating a coordinate matrix.
18) FSCRT_Matrix_Scale: used for scaling a coordinate matrix.
19) FSCRT_Matrix_Shear: used for making a shear transformation on a coordinate matrix.
20) FSPDF_Doc_Close: used for closing a PDF document.
21) FSCRT_File_Release: used for releasing a file object.

3.28.3 Run

Please refer to the 3.1.3 to run this demo.

3.29 Annots_summary

3.29.1 Description

1) Purpose: illustrates how to use Foxit PDF SDK to summarize the annotations in a PDF document.
2) Property: this demo is a console application without UI.

   Usage: *annots_summary*
3) Process: this demo loads specific PDF file from input directory. For the PDF file, this demo will traverse the annotations in a PDF document, and then create an annotation summary which will be saved to the output directory. Once completed, the demo will exit. All the information about execution process will be recorded into log file.
4) Input directory: used to place input files.

   Default: /samples/simple_sample/input_files.

   Use default files: Annot_all.pdf.
5) Output directory: used to store output files, including log.txt.

   Default: /samples/simple_sample/output_files/annots_summary.

   Default output files: annots_summary_commentsonly.pdf and annots_summary.pdf.
6) Program files:

   pdflayer.cpp: source code of summarizing the annotations in a PDF document.

3.29.2 SDK functions

The demo of matrix uses following additional functions of Foxit PDF SDK:

1) FSCRT_PDFModule_Initialize: used for initializing PDF module.
2) FSCRT_PDFModule_Finalize: used for finalizing PDF module.
3) FSPDF_Doc_StartLoad: used for loading a PDF file.
4) FSPDF_Doc_CountPages: used for counting the number of pages of a PDF file.
5) FSPDF_Doc_Create: used for creating a new document object
6) FSPDF_Doc_GetPage: used for getting a page object.
7) FSCRT_Font_Create: used for creating a font with the given attributes.
8) FSPDF_Page_Create: used for creating a new page to a position specified by the index.
9) FSPDF_Page_GetSize: used for getting a page size.
10) FSPDF_Page_StartParse: used for starting parsing a PDF page.

11) FSPDF_Page_LoadAnnots: used for load annotations from a PDF page.
12) FSPDF_Page_SetSize: used for setting page size.
13) FSPDF_Page_GetPageObjects: used for getting page objects in a PDF page.
14) FSPDF_FormXObject_Create: used for creating a Form XObject object.
15) FSPDF_FormXObject_ExtractAPFromPage: used for extracting appearance of contents and annotations in a source page to a Form XObject object.
16) FSPDF_PageObject_SetMatrix: used for setting matrix of a page object.
17) FSPDF_PageObjects_InsertObject: used for inserting a page object and it will be automatically freed.
18) FSPDF_PathObject_Create: used for creating a new path object.
19) FSPDF_PathObject_SetPathData: used for setting path data of a path object.
20) FSPDF_PageObject_SetColor: used for setting color of a page object.
21) FSCRT_PathData_Create: used for creating a new path data.
22) FSCRT_PathData_MoveTo: used for adding a point to start a figure. This point will become a new current position.
23) FSCRT_PathData_LineTo: used for adding a point to a given figure, and a line is to be drawn from the current position to this point, which then becomes a new current position.
24) FSCRT_PathData_Clear: used for clearing all points of the given path data.
25) FSCRT_PathData_Release: used for releasing all related resources of the given path data.
26) FSCRT_PathData_CountPoints: used for getting number of points of the given path data.
27) FSCRT_PathData_GetPoint: used for getting a specific point of the path data.
28) FSPDF_TextObject_Create: used for creating a new text object.
29) FSPDF_TextObject_SetTextState: used for setting text states of a text object.
30) FSPDF_TextObject_SetUnicodeString: used for setting Unicode string for a text object.
31) FSPDF_TextObject_Create: used for creating a new text object.
32) FSPDF_Annot_GetCount: used for getting count of annotations, by specific filter.
33) FSPDF_Annot_Get: used for getting annotation with specified index, by specific filter.
34) FSPDF_Annot_GetRect: used for getting rectangle of an annotation.
35) FSPDF_Annot_GetType: used for getting type of an annotation.

36) FSPDF_Annot_GetTitle: used for getting title of a markup annotation.
37) FSPDF_Annot_GetSubject: used for getting subject of a markup annotation.
38) FSPDF_Annot_GetModifiedDateTime: used for getting modification time of an annotation.
39) FSPDF_Annot_GetContents: used for getting contents of an annotation.
40) FSPDF_Annot_GetDefaultAppearance: used for getting default appearance of a free text annotation, which can be used in formatting text.
41) FSPDF_Annot_GetVertices: used for getting vertices of a polygon or polyline annotation.
42) FSPDF_Annot_GetInkList: used for getting ink list data of an ink annotation.
43) FSPDF_Annot_GetIntent: used for getting intent of a markup annotation.
44) FSPDF_Annot_GetCalloutLinePoints: used for getting callout line points of a free text annotation.
45) FSCRT_Image_LoadFromFile: used for loading an image from an image file.
46) FSCRT_Image_CountFrames: used for counting frames of an image.
47) FSCRT_Image_LoadFrame: used for loading an image frame by index.
48) FSPDF_ImageObject_Create: used for creating an image object.
49) FSPDF_ImageObject_SetImage: used for setting an image to an image object.

3.29.3 Run

Please refer to the 3.1.3 to run this demo.

3.30 Pdfdigitalsignature

3.30.1 Description

1) Purpose: illustrates how to use Foxit PDF SDK to add and verify signatures with a time stamp in PDF files.
2) Property: this demo is a console application without UI, which relies on the third-part library OpenSSL.

   It supports command line argument.

   Usage: *pdfdigitalsignature [/i [inputfile] /sf [subfilter] /o [destfolder] /t [sig/doc] /u [url] /up [user:pswd] /p [port]]*

/i [inputfile]	Path of a PDF file as input file which will be inserted a time stamp.
/sf [subfilter]	The subfilter of signature. It can be set to "adbe.pkcs7.sha1", "adbe.pkcs7.detached", or "ETSI.RFC3161".
/o [destfolder]	Path of a folder as output directory. It cannot be same with the prefix of the input file. If not set, use default output directory.
/t [sig/doc]	The value “Sig” means adding a time stamp to a signature, and “doc” means adding a time stamp to a document.
/u [url]	The url of TSA server. It is required if /t is used.
/up [user:pswd]	User name and password for TSA server.
/p [port]	TSA server port.

3) Process: this demo loads specific PDF files from input directory. For each PDF file, this demo will load the first page of the PDF document, create and sign digital signatures with a time stamp for the PDF document, and then verify the signatures. This demo will sign the document twice. The signed PDF will be saved as a new one to output directory. Once completed, the demo will exit. All the information about execution process will be recorded into log file.
4) Input directory: used to place input files.
   a) Default: /samples/simple_sample/input_files.

      Use default files: AboutFoxit.pdf.
   b) User-defined: set single input file through command line argument.
5) Output directory: used to store output files, including log.txt.
   a) Default: /samples/simple_sample/output_files/pdfdigitalsignature.

   b) User-defined: set path of output folder through command line argument and output directory will be “/outputfolder/pdfdigitalsignature”.

6) Certificate: This demo uses the default certificate “foxit_all.pfx” under “sampls/simple_sample/input_files” folder. Generally, a certificate is encrypted with a password. The password should be changed if users change the encryption certificate. In this demo, the password is located in line 123 in “param.cpp”.

7) Program files:

pdfdigitalsignature.cpp:	source code of adding and verifying signatures in PDF files.
param.cpp：	source code of initializing and validating the parameters.
pkcs7_sign.cpp:	source code of using OpenSSL to sign the data and time stamp.

3.30.2 SDK functions

The demo of pdfdigitalsignature uses following additional functions of Foxit PDF SDK:

1) FSCRT_PDFModule_Initialize: used for initializing PDF module.
2) FSCRT_PDFModule_Finalize: used for finalizing PDF module.
3) FSPDF_Doc_StartLoad: used for loading a PDF file.
4) FSPDF_Doc_GetPage: used for getting a page object.
5) FSPDF_Doc_GetSignature: used for getting a signature object.
6) FSPDF_Doc_Close: used for closing a PDF document.
7) FSPDF_Page_Clear: used for releasing all page contents and relative resources.
8) FSCRT_Image_LoadFromFile: used for loading an image from an image file.
9) FSCRT_Image_LoadFrame: used for loading an image frame by index.
10) FSCRT_Image_GetCurrentFrameBitmap: used for retrieving the bitmap of the current frame.
11) FSCRT_Image_Release: used for releasing an image object.
12) FSCRT_Bitmap_Release: used for destroying a bitmap and release related resources.
13) FSPDF_Signature_Add: used for adding an unsigned signature field with blank appearance to a specific position in a PDF page.
14) FSPDF_Signature_InitValue: used for initializing a signature field.
15) FSPDF_Signature_SetFilter: used for setting the name of the preferred signature handler to use for signature.
16) FSPDF_Signature_SetSubFilter: used for setting sub filter of a signature.

17) FSPDF_Signature_SetSigner: used for setting signer name of a signature.
18) FSPDF_Signature_SetText: used for setting the text content out of signature appearance.
19) FSPDF_Signature_SetLocation: used for setting the CPU host name or physical location of the signing
20) FSPDF_Signature_SetReason: used for setting signing reason of a signature.
21) FSPDF_Signature_SetDateTime: used for setting the creation time of a signature.
22) FSPDF_Signature_SetDistinguishedName: used for setting distinguished name of signer in a signature certificate.
23) FSPDF_Signature_SetAppearanceFlags: used for setting signature appearance flags required for display appearance.
24) FSPDF_Signature_SetBitmap: used for setting a bitmap to icon in signature display.
25) FSPDF_Signature_ResetAppearance: used for resetting the appearance of a signature field.
26) FSPDF_Signature_RegisterHandler: used for registering a third-party handler to Foxit PDF SDK, along with its filter and sub filter.
27) FSPDF_Signature_StartSign: used for signing a PDF document using custom signature progressively.
28) FSPDF_Signature_StartVerify: used for verifying a signature by custom way in progressive manner.

3.30.3 Run

Please refer to the 3.1.3 to run this demo.

3.30.4 Note

- The Signature module only provides the third-party signature interface and requires that the customers have their own signature implementation. If you want to purchase Foxit PDF SDK license and use any functions of this module, please contact Foxit to enable this module explicitly.
- Before running this demo, please make sure that you have already installed OpenSSL1.0.0q or later. For MAC OS X, please install OpenSSL with 64-bit, not 32-bit.

  In addition, please use the corresponding compiled OpenSSL libraries for your system, because the current system may not be compatible with the OpenSSL libraries that are compiled by other systems. For example, if you run the demo in Windows 10, you had better use the OpenSSL libraries which are compiled in Windows 10 platform.

On Windows, If you run this demo in VC++ 6.0, please change the path of loading header files and libraries: click “Tools -> Options -> Directory -> Show directories for”, add the OpenSSL folder path and libraries path to “include files” and “library files” respectively. Note: in the adding list, move the adding path to the top which can ensure that the external references header files and libraries could be searched first.

On MAC OS X and Linux, you can also download an OpenSSL source package, for example: openssl-1.0.0q.tar.gz, and then configure prefix path to /usr/local. The default openssl path is /usr/local/ssl in Makefile file, if you change the openssl installation path, please update Makefile to new path when executing "make pdfdigitalsignature" command.

On iOS, you should put openssl package into the directory “lib” of foxit pdf sdk package. The default paths of Xcode configuration of the demo are "$(SRCROOT)/../../../lib/ssl/lib" and "$(SRCROOT)/../../../lib/ssl/include”. If you change the openssl package path, please update the Xcode configuration.

3.31 Addformfields

3.31.1 Description

1) Purpose: illustrates how to use Foxit PDF SDK to add form fields to a PDF document.
2) Property: this demo is a console application without UI.

   It supports command line argument.

   Usage: *addformfields*
3) Process: this demo loads specific PDF file from input directory. For the PDF file, this demo will create some form fields, such as text field, radio button, combo box, check box, list box and push button, set the values and properties of those form fields, and then add them to the source PDF document which will be saved to the output directory. Once completed, the demo will exit. All the information about execution process will be recorded into log file.
4) Input directory: used to place input files.

   Default: /samples/simple_sample/input_files.

   Use default files: test_addFormFields.pdf.
5) Output directory: used to store output files, including log.txt.

   Default: /samples/simple_sample/output_files/addformfields.

   Default output files: addFormFields.pdf.
6) Program files:

   addformfields.cpp: source code of adding form fields to a PDF file.

3.31.2 SDK functions

The demo of addformfields uses following additional functions of Foxit PDF SDK:

1) FSCRT_PDFModule_Initialize: used for initializing PDF module.
2) FSCRT_PDFModule_Finalize: used for finalizing PDF module.
3) FSPDF_Doc_StartLoad: used for loading a PDF file.
4) FSPDF_Doc_Close: used for closing a PDF document.
5) FSPDF_Doc_GetPage: used for getting a page object.
6) FSPDF_Doc_CreateForm: used for creating an arco form.
7) FSPDF_Page_LoadAnnots: used for loading annotations from a PDF page.
8) FSPDF_Page_Clear: used for releasing all page contents and relative resources.

9) FSPDF_Form_AddField: used for adding a form field to arco form.
10) FSPDF_FormField_SetValue: used for setting value of a field (except signature field).
11) FSPDF_FormControl_SetChecked: used for setting check box or radio button's state.
12) FSPDF_FormControl_SetExportValue: used for setting option's export value.
13) FSPDF_FormField_SetOptions: used for setting list box or combo box options.
14) FSPDF_FormControl_GetWidgetAnnot: used for getting widget annotation from form control handle.
15) FSPDF_Annot_SetHighlightingMode: used for setting highlighting mode of a link or widget annotation.
16) FSPDF_Annot_SetMKColor: used for setting color of specific type in MK dictionary.
17) FSPDF_Annot_SetMKCaption: used for setting caption of a specific type in MK dictionary.
18) FSPDF_Annot_ResetAppearance: used for resetting (regenerating) appearance of an annotation.
19) FSPDF_Attachment_Create: used for creating an attachment object.
20) FSPDF_Attachment_SetFileName: used for setting the file name of an attachment.
21) FSPDF_Annot_InsertAction: used for inserting an action of specified index associated with an annotation.
22) FSPDF_Page_UnloadAnnots: used for unloading annotations of a PDF page.

3.31.3 Run

Please refer to the 3.1.3 to run this demo.

3.32 Layeredit

3.32.1 Description

1) Purpose: illustrates how to use Foxit PDF SDK to add and edit layers in a PDF document.
2) Property: this demo is a console application without UI.

   It supports command line argument.

   Usage: *layeredit*
3) Process: this demo loads specific PDF file from input directory. For the PDF file, this demo will create four layers named “Image”, “Path”, “Text” and “Other” respectively, and then move all of the image/path/text objects of the PDF file to the corresponding layers, and the other objects will be moved to “Other” layer. It also realizes the features like setting/removing the application usage of the PDF layers, removing the page objects of the PDF layers, and removing the layers of the PDF file. The operated PDF documents will be saved to the output directory. Once completed, the demo will exit. All the information about execution and API calling process will be recorded into log files.
4) Input directory: used to place input files.

   Default: /samples/simple_sample/input_files. Here’re two ways:

   i. Use default files: FoxitBigPreview.pdf.
   ii. Set default sub-folder: create a sub-folder named “layeredit” and put all input files into this sub-folder. When running, this demo will enumerate files in sub-folder as input files.
5) Output directory: used to store output files, including log.txt and gsdkLog.txt.

   Default: /samples/simple_sample/output_files/layeredit.

   Default output files: FoxitBigPreview.pdf, AddLayerUsage_FoxitBigPreview.pdf, RemoveLayers_FoxitBigPreview.pdf, RemoveLayerUsage_FoxitBigPreview.pdf, and RemoveObjects_FoxitBigPreview.pdf.
6) Program files:

   layeredit.cpp: source code of accessing and editing layer properties of PDF documents.

3.32.2 SDK functions

The demo of layeredit uses following additional functions of Foxit PDF SDK:

1) FSCRT_PDFModule_Initialize: used for initializing PDF module.
2) FSCRT_PDFModule_Finalize: used for finalizing PDF module.
3) FSPDF_Doc_StartLoad: used for loading a PDF file.
4) FSPDF_Doc_GetPage: used for getting a page object.
5) FSPDF_Doc_EnumLayers: used for enumerating all PDF layers.
6) FSPDF_Doc_CountPages: used for getting the page count of a document.
7) FSPDF_Doc_StartSaveToFile: used for starting an incremental saving progress for PDF files.
7) FSPDF_Doc_AddLayer: used for adding a layer to a PDF document.
8) FSPDF_Doc_SetLayersBaseState: used for setting the base visible state of PDF layers in the default configuration.
9) FSCRT_Array_Init: used for initializing array.
10) FSPDF_Page_StartParse: used for starting parsing a PDF page.
11) FSPDF_Page_Clear: used for releasing all page contents and relative resources.
12) FSPDF_Doc_Close: used for closing a PDF document.
13) FSPDF_PageObject_GetType: used for getting type of a page object.
14) FSPDF_PageObjects_EnumerateObject: used for enumerating page objects and retrieving a page object at a given position from page objects.
15) FSPDF_PageObject_GetLayers: used for getting all the layers which associate with specific page object.
16) FSPDF_PageObjects_GenerateContents: used for generating PDF Page contents.
17) FSPDF_Page_GetPageObjects: used for getting page objects in a PDF page.
18) FSPDF_Layer_AddPageObject: used for adding a page object to a PDF layer.
19) FSPDF_Layer_RemovePageObject: used for removing a page object from a PDF layer.
20) FSPDF_Layer_Remove: used for removing a layer from the document.
21) FSPDF_LayerNode_Init: used for initializing a PDF layer node.
22) FSPDF_Layer_EnumeratePageObject: used for enumerating page objects and retrieving a page object which is related with a specific layer.
23) FSPDF_LayerNode_Clear: used for clearing a PDF layer node.
24) FSPDF_Layer_SetApplicationUsage: used for setting the application usage of a PDF layer.
25) FSPDF_Layer_GetApplicationUsage: used for getting the application usage and it's

param of a PDF layer.

26) FSPDF_Layer_RemoveApplicationUsage: used for removing the application usage of a PDF layer.

3.32.3 Run

Please refer to the 3.1.3 to run this demo.