Developer Guide

Foxit PDF SDK

Microsoft® Partner

Gold Independent Software Vendor (ISV)

TABLE OF CONTENTS

1 INTRODUCTION TO FOXIT PDF SDK

Have you ever thought about building your own application that can do everything you want with PDF files? If your answer is "Yes", congratulations! You just found the best solution in the industry that allows you to build stable, secure, efficient and full-featured PDF applications.

Foxit PDF SDK provides high-performance libraries to help any software developer add robust PDF functionality to their enterprise, mobile and cloud applications across all platforms (includes Windows, Mac, Linux, Web, Android, iOS, and UWP), using the most popular development languages and environments.

1.1 Why Foxit PDF SDK is your choice

Foxit is a leading software provider of solutions for reading, editing, creating, organizing, and securing PDF documents. Foxit PDF SDK libraries have been used in many of today's leading apps, and they are proven, robust, and battle-tested to provide the quality, performance, and features that the industry's largest apps demand. Customers choose Foxit PDF SDK product for the following reasons:

- **Easy to integrate**

Developers can seamlessly integrate Foxit PDF SDK into their own applications.

- **Lightweight footprint**

Do not exhaust system resource and deploys quickly.

- **Cross-platform support**

Support current mainstream platforms, such as Windows, Mac, Linux, Web, Android, iOS, and UWP.

- **Powered by Foxit's high fidelity rendering PDF engine**

The core technology of the SDK is based on Foxit's PDF engine, which is trusted by a large number of the world's largest and well-known companies. Foxit's powerful engine makes the app fast on parsing, rendering, and makes document viewing consistent on a variety of devices.

- **Premium World-side Support**

Foxit offers premium support for its developer products because when you are developing mission critical products you need the best support. Foxit has one of the PDF industry's largest team of support engineers. Updates are released on a regular basis to improve user experience by adding new features and enhancements.

1.2 Foxit PDF SDK for C++ API

Application developers who use Foxit PDF SDK can leverage Foxit's powerful, standard-compliant PDF technology to securely display, create, edit, annotate, format, organize, print, share, secure, search documents as well as to fill PDF forms. Additionally, Foxit PDF SDK (for C++ and .NET) includes a built-in, embeddable PDF Viewer, making the development process easier and faster. For more detailed information, please visit the website https://developers.foxitsoftware.com/pdf-sdk/.

In this guide, we focus on the introduction of Foxit PDF SDK for C++ API on Windows, Linux and Mac platforms.

Foxit PDF SDK for C++ API ships with simple-to-use APIs that can help C++ developers seamlessly integrate powerful PDF technology into their own projects on Windows, Linux and Mac platforms. It provides rich features on PDF documents, such as PDF viewing, bookmark navigating, text selecting/copying/searching, PDF signatures, PDF forms, rights management, PDF annotations, and full text search.

1.3 Evaluation

Foxit PDF SDK allows users to download trial version to evaluate SDK. The trial version has no difference from a standard version except for the 10-day limitation trial period and the trail watermarks that will be generated on the PDF pages. After the evaluation period expires, customers should contact Foxit sales team and purchase licenses to continue using Foxit PDF SDK.

1.4 License

Developers should purchase licenses to use Foxit PDF SDK in their solutions. Licenses grant users permissions to release their applications based on PDF SDK libraries. However, users are prohibited to distribute any documents, sample codes, or source codes in the SDK released package to any third party without the permission from Foxit Software Incorporated.

1.5 About this guide

This guide is intended for the developers who need to integrate Foxit PDF SDK with C++ program language into their own applications. It aims at introducing the installation package, and the usage of SDK.

2 GETTING STARTED

It's very easy to setup Foxit PDF SDK and see it in action! This guide will provide you with a brief introduction about our SDK package. As a cross-platform product, Foxit PDF SDK supports the identical interfaces for desktop system of Windows, Linux, and Mac. The following sections introduce the contents of system requirements, the installation package as well as how to run a demo, and create your own project.

2.1 System Requirements

Platform	**System Requirement**	**Memo**
Windows	Windows XP, Vista, 7, 8 and 10 (32-bit and 64-bit) Windows Server 2003, 2008 and 2012 (32-bit and 64-bit)	It only supports for Windows 8/10 classic style, but not for Store App or Universal App.
Linux	32-bit and 64-bit OS	All Linux samples have been tested on Ubuntu14.0 32/64 bit.
Mac	Mac OS X 10.6 or higher	

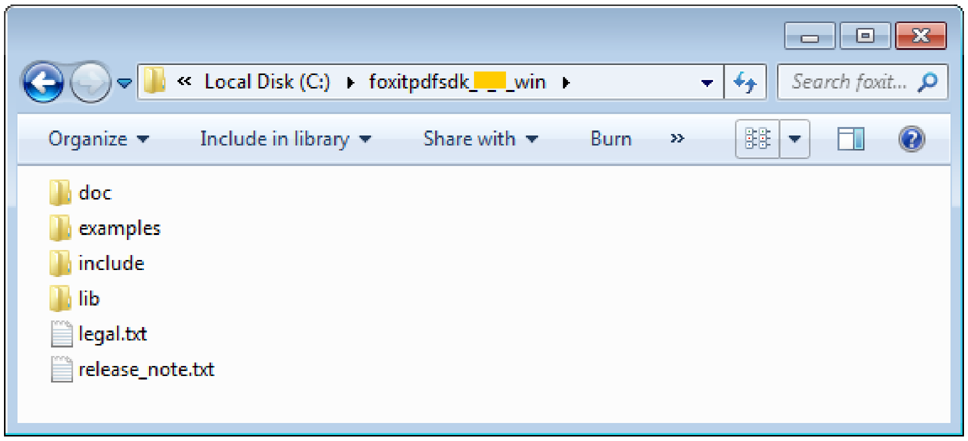
2.2 Windows

In this guide, one thing to note is that the highlighted rectangle in the figure is the version of the SDK. Here the SDK version is 7.3, so it shows 7_3.

2.2.1 What is in the package

Download Foxit PDF SDK zip for Windows package and extract it to a new directory "foxitpdfsdk_7_3_win", which is shown in Figure 2-1. The release package contains the following folders:

doc:	API references, developer guide
examples:	sample projects and demos
include:	header files for Foxit PDF SDK API
lib:	libraries and license files



doc
examples
include
lib
legal.txt
release_note.txt

Figure 2-1

In the "examples" folder, there are two types of demos. "\examples\simple_demo" contains more than 30 demos that cover a wide range of PDF applications. "\examples\view_demo" contains a UI demo that realizes a lite PDF viewer.

2.2.2 How to run a demo

Simple Demo

Simple demo projects provide examples to show developers about how to effectively apply Foxit PDF SDK APIs to complete their applications.

To run a demo in Visual Studio (except security, signature, ocr, compliance, html2pdf and office2pdf demos which will be introduced later), you can follow the steps below:

1) Load the visual studio solution files "simple_demo_vs2010.sln" or "simple_demo_vs2015.sln" or "simple_demo_vs2017.sln" (depending on your Visual Studio version) in the "\examples\simple_demo" folder.

2) Build all the demos by clicking "Build > Build Solution". Alternatively, if you merely want to build a specific demo, you can right-click it and then choose "Build" or load the "*.vxcproj" file in the folder of a specific demo project and then build it.

   After building, the executable file ".exe" will be generated in the "\examples\simple_demo\bin" folder. And the names of the executable files depend on the build configurations.

3) Run a specific executable file, just double-click it.

   Some demos will generate output files (pdf, text or image files) to a folder named by the project name under "examples\simple_demo\output_files\" folder.

> ***Note**: If you want to see the detailed executing processes, you can run it in command line. Start "cmd.exe", navigate to "\examples\simple_demo\bin", and run a specific executable file.*

Security demo

Before running **security** demo, you should install the certificates "foxit.cer" and "foxit_all.pfx" found in "\examples\simple_demo\input_files" folder.

a) To install "foxit.cer", double-click it to start the certificate import wizard. Then select "Install certificate… > Next > Next > Finish".

b) To install "foxit_all.pfx", double-click it to start the certificate import wizard. Then select "Next > Next > (Type the password for the private key in the textbox) and click Next > Next > Finish".

c) Run the demo following the steps as the other demos.

Signature demo

Before running **signature** demo, you should ensure that the OpenSSL has been already installed in your machine. Download an OpenSSL source package from the OpenSSL website, or you can contact us directly, and then extract it and do the following:

1) Put the OpenSSL folder into the "include" folder which ensures that the OpenSSL header files included in the demos can be found.

2) Put the "libeay32.lib" library into the "lib" folder.

3) Run the demo following the steps as the other demos.

***Note**: We have verified that OpenSSL 1.0.2m version is available on the signature demo, you can replace it with the desired version, and maybe need to do some changes.*

OCR and Compliance demos

For **ocr** and **compliance** demos, you should build a resource directory at first, please contact Foxit support team or sales team to get the resource files packages. For more details about how to run the demos, please refer to section 3.34 "OCR" and section 3.35 "Compliance".

HTML to PDF demo

For html2pdf demo, you should contact Foxit support team or sales team to get the engine files package for converting from HTML to PDF at first. For more details about how to run the demo, please refer to section 3.37 "HTML to PDF Conversion".

Office to PDF demo

For office2pdf demo, you should make sure that Microsoft Office 2007 version or higher is already installed and the default Microsoft virtual printer is already set on your Windows system. Then, run the demo following the steps as the other demos.

View Demo

This view demo provides an example for developers to realize a PDF reader using Foxit PDF SDK APIs.

To run the demo in Visual Studio, load "PDFReader_VS2010.sln" or "PDFReader_VS2015.sln" or "PDFReader_VS2017.sln " (depending on your Visual Studio version) in the "\examples\view_demo\PDFReader\project" folder, and then click "Debug > Start Without Debugging" to run it. After the demo starts, you will see the following window:

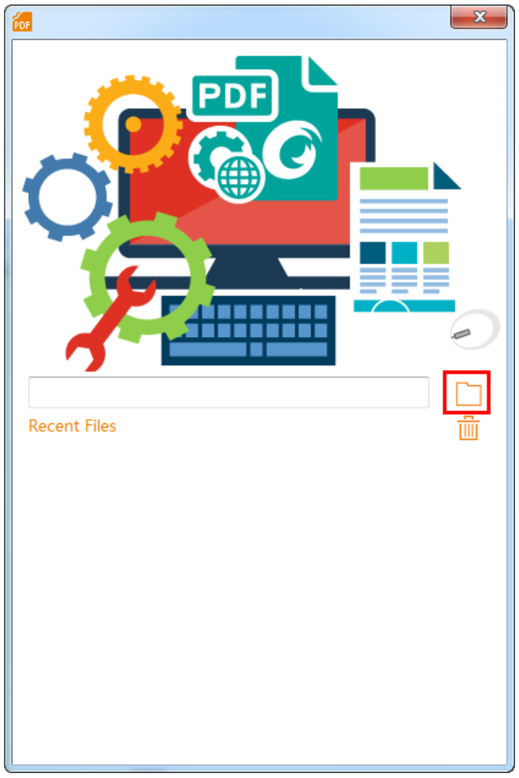
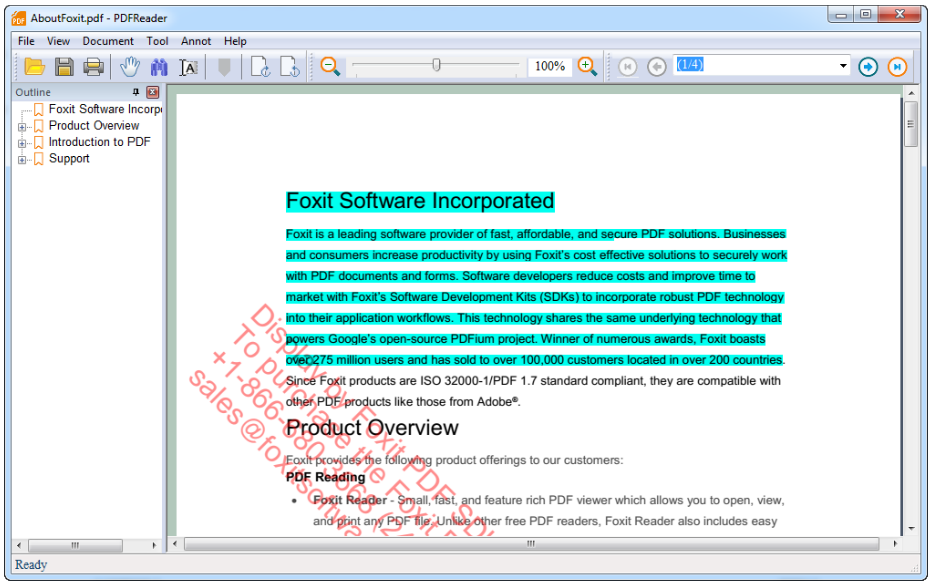


Figure 2-2

You can drag a PDF file to the above window (Figure 2-2) to open it and browse the content by scrolling down or moving the PDF page by holding the left mouse button. Besides, you can click "Annot > HighLight" to highlight the selected text. A screenshot of the demo is shown in Figure 2-3.



Foxit Software Incorporated

Foxit is a leading software provider of fast, affordable, and secure PDF solutions. Businesses and consumers increase productivity by using Foxit's cost effective solutions to securely work with PDF documents and forms. Software developers reduce costs and improve time to market with Foxit's Software Development Kits (SDKs) to incorporate robust PDF technology into their application workflows. This technology shares the same underlying technology that powers Google's open-source PDFium project. Winner of numerous awards, Foxit boasts over 275 million users and has sold to over 100,000 customers located in over 200 countries. Since Foxit products are ISO 32000-1/PDF 1.7 standard compliant, they are compatible with other PDF products like those from Adobe®.

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

Figure 2-3

2.2.3 How to create a simple project

In this section, we will show you how to use Foxit PDF SDK for Windows to create a simple project that renders the first page of a PDF to a bitmap and saves it as a JPG image. Please follow the steps below:

1) Open Visual Studio and create a new Win32 Console Application named "test_win".

2) Copy the "include" and "lib" folders from the "foxitpdfsdk_7_3_win" folder to the project "test_win" folder.

3) Add the "include" folder to your "Additional Include Directories". Right-click the *test_win* project in Solution Explorer, choose "Properties", and find "Configuration Properties > C/C++ > General > Additional Include Directories".

4) Add include header statements to the beginning of test_win.cpp.

```
#include <iostream>
#include "../include/common/fs_common.h"
#include "../include/pdf/fs_pdfdoc.h"
#include "../include/pdf/fs_pdfpage.h"
#include "../include/common/fs_render.h"
```

5) Add using namespace statements.

```
using namespace std;
using namespace foxit;
using namespace common;
using namespace pdf;
using namespace foxit::common;
```

6) Include Foxit PDF SDK library.

```
#if defined (_WIN64) // windows 64bit platforms.
        #if defined (_DEBUG)
                #pragma comment (lib, "../lib/fsdk_win64.lib")
        #else
                #pragma comment (lib, "../lib/fsdk_win64.lib")
        #endif

#elif defined (_WIN32) // windows 32bit platforms.
        #if defined (_DEBUG)
                #pragma comment (lib, "../lib/fsdk_win32.lib")
        #else
                #pragma comment (lib, "../lib/fsdk_win32.lib")
        #endif
#endif
```

7) Initialize the Foxit PDF SDK library. It is necessary for apps to initialize Foxit PDF SDK using a license before calling any APIs. The trial license files can be found in the "lib" folder.

```
const char* sn = " ";
const char* key = " ";
foxit::ErrorCode code = Library::Initialize(sn, key);
if (code != foxit::e_ErrSuccess) {
        return FALSE;
}
```

Note *The value of "sn" can be got from "**gsdk_sn.txt**" (the string after "SN=") and the value of "key" can be got from "**gsdk_key.txt**" (the string after "Sign=").*

8) Load a PDF document, and parse the first page of the document. Let us assume that you have already put a "Sample.pdf" to the "test_win\test_win" folder.

```
PDFDoc doc("Sample.pdf");
ErrorCode error_code = doc.Load();
if (error_code!= foxit::e_ErrSuccess) return 0;
PDFPage page = doc.GetPage(0);
page.StartParse(foxit::pdf::PDFPage::e_ParsePageNormal, NULL, false);
```

9) Render a Page to a bitmap and save it as a JPG file.

```
int width = static_cast<int>(page.GetWidth());
int height = static_cast<int>(page.GetHeight());
Matrix matrix = page.GetDisplayMatrix(0, 0, width, height, page.GetRotation());

// Prepare a bitmap for rendering.
Bitmap bitmap(width, height, Bitmap::e_DIBArgb, NULL, 0);
bitmap.FillRect(0xFFFFFFFF, NULL);
// Render page.
Renderer render(bitmap, false);
render.StartRender(page, matrix, NULL);

// Add the bitmap to image and save the image.
Image img;
img.AddFrame(bitmap);
img.SaveAs("testpage.jpg");
```

10) Click "Build > Build Solution" to build the project. The executable file "test_win.exe" will be generated in "test_win\Debug" or "test_win\Release" folder depending on the build configurations.

11) Copy "fsdk_win32.dll" or "fsdk_win64.dll" in the "lib" folder to the output directory ("test_win\Debug" or "test_win\Release"). Please make sure that the "fsdk_win**.dll" architecture needs to match the platform target (Win32 or Win64) of the application.

12) Run the project. Choose one of the following:

    i. Click "Debug > Start Without Debugging" in Visual Studio to run the project, and the "testpage.jpg" will be generated in the "test_win\test_win" folder (same with "test_win.cpp").

    ii. Double-click the executable file "test_win.exe" to run the project. In this way, you should put the "Sample.pdf" to the same folder with the "test_win.exe", and the "testpage.jpg" will also be generated in the same folder.

The final contents of "test_win.cpp" is as follow:

```
#include "stdafx.h"

#include <iostream>
#include "../include/common/fs_common.h"
```

```
#include "../include/pdf/fs_pdfdoc.h"
#include "../include/pdf/fs_pdfpage.h"
#include "../include/common/fs_render.h"

using namespace std;
using namespace foxit;
using namespace common;
using namespace pdf;
using namespace foxit::common;

// Include Foxit PDF SDK library.
#if defined (_WIN64) // windows 64bit platforms.
	#if defined (_DEBUG)
		#pragma comment (lib, "../lib/fsdk_win64.lib")
	#else
		#pragma comment (lib, "../lib/fsdk_win64.lib")
	#endif

#elif defined (_WIN32) // windows 32bit platforms.
	#if defined (_DEBUG)
		#pragma comment (lib, "../lib/fsdk_win32.lib")
	#else
		#pragma comment (lib, "../lib/fsdk_win32.lib")
	#endif
#endif

int _tmain(int argc, _TCHAR* argv[])

{
	// The value of "sn" can be got from "gsdk_sn.txt" (the string after "SN=").
	// The value of "key" can be got from "gsdk_key.txt" (the string after "Sign=").
	const char* sn = " ";
	const char* key = " ";
	foxit::ErrorCode code = Library::Initialize(sn, key);
	if (code != foxit::e_ErrSuccess) {
		return FALSE;
	}

	// Load a PDF document, and parse the first page of the document.
	PDFDoc doc("Sample.pdf");
```

```
        ErrorCode error_code = doc.Load();
        if (error_code!= foxit::e_ErrSuccess) return 0;
        PDFPage page = doc.GetPage(0);
        page.StartParse(foxit::pdf::PDFPage::e_ParsePageNormal, NULL, false);

        int width = static_cast<int>(page.GetWidth());
        int height = static_cast<int>(page.GetHeight());
        Matrix matrix = page.GetDisplayMatrix(0, 0, width, height, page.GetRotation());

        // Prepare a bitmap for rendering.
        Bitmap bitmap(width, height, Bitmap::e_DIBArgb, NULL, 0);
        bitmap.FillRect(0xFFFFFFFF, NULL);
        // Render page.
        Renderer render(bitmap, false);
        render.StartRender(page, matrix, NULL);

        // Add the bitmap to image and save the image.
        Image img;
        img.AddFrame(bitmap);
        img.SaveAs("testpage.jpg");
        return 0;
}
```

2.3 Linux

In this guide, one thing to note is that the highlighted rectangle in the figure is the version of the SDK. Here the SDK version is 7.3, so it shows 7_3.

2.3.1 What is in this package

Download Foxit PDF SDK zip for Linux package and extract it to a new directory "foxitpdfsdk_7_3_linux".The structure of the release package is shown in Figure 2-4. This package contains the following folders:

doc:	API references, developer guide
examples:	sample projects and demos
include:	header files for Foxit PDF SDK API
lib:	libraries and license files

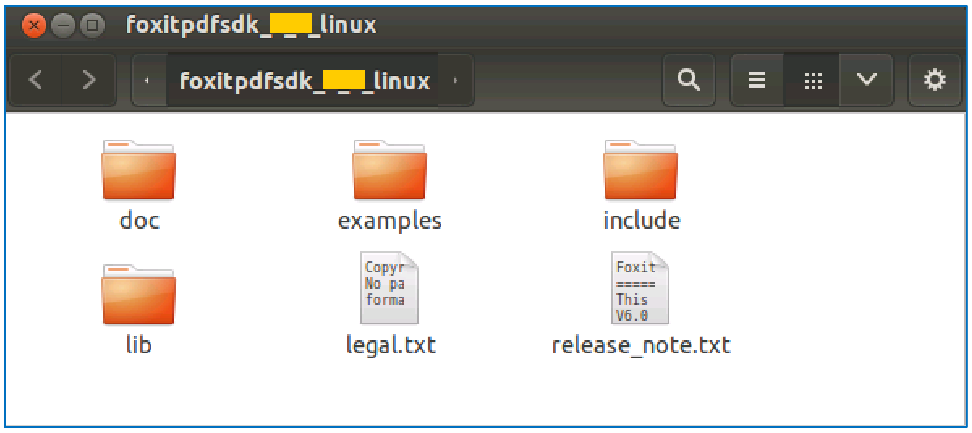


Figure 2-4

In the "examples" folder, there are two types of demos. "\examples\simple_demo" contains a wide range of PDF document application demos. "\examples\view_demo" contains a Qt UI demo that realizes a lite PDF viewer.

2.3.2 How to run a demo

Simple Demo

Simple demo projects provide examples to show developers about how to effectively apply Foxit PDF SDK APIs to complete their applications.

Before running the demos, please make sure you have configured the environment correctly and installed CMake (3.1 or later) on your machine.

OS	Compiler tools or IDE	Version
Linux	GCC	4.8 and later

To run a demo in a terminal window (except security, signature, and compliance demos which will be introduced later), please follow the steps:

1) Open a terminal window, navigate to "foxitpdfsdk_7_3_linux\examples\simple_demo";

2) Run "**cmake -DPRJ_NAME=XXX**" to compile a specific demo. "XXX" is the demo name, which must be "annotation", "attachment", "pdf2text" and so on. For example, run "cmake -DPRJ_NAME=annotation".

3) Run "**make**" to build the demo specified in the above command. Then the executable file will be generated named "**XXX_xxx**". "XXX" is the demo name and "xxx" is the architecture name, such as "annotation_linux64".

4) Run "**./XXX_xxx**" to run the demo. For example, run "./annotation_linux64" to run the annotation demo.

Some demos will generate output files (pdf, text or image files) to a folder named by the project name under "examples/simple_demo/output_files/".

Security and Signature demos

Before running **security** and **signature** demos, please make sure that you have already installed OpenSSL. Download an OpenSSL source package from the OpenSSL website, or you can contact us directly, and then extract it and do the following:

1) Put the OpenSSL folder into the "include" folder which ensures that the OpenSSL header files included in the demos can be found.
2) Put the "libssl.a" and "libcrypto.a" libraries into the "lib" folder.
3) Run the demos following the steps as the other demos.

***Note**: We have verified that OpenSSL 1.0.2m version is available on the security and signature demos, you can replaced it with the desired version, and maybe need to do some changes.*

Compliance demo

For how to run the **compliance** demo, please refer to section 3.35 "Compliance".

View Demo

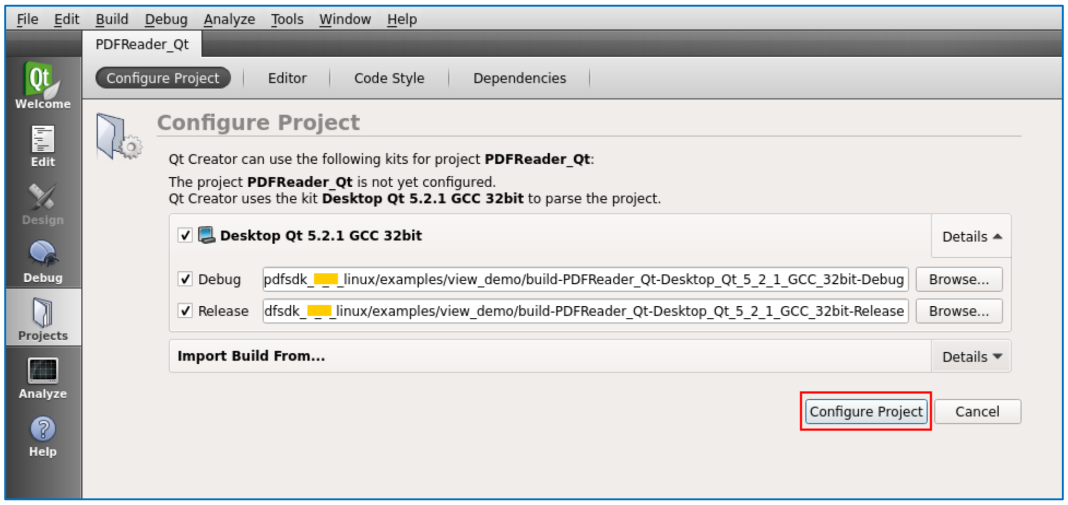
This view demo provides a Qt example for developers to realize a PDF reader using Foxit PDF SDK APIs.

Before running the demo, please make sure that the compiler tool (GCC 4.8 or later) and the IDE "Qt Creator" are available on your Linux system.

To run the demo in Qt Creator, please follow the steps below:

1) Open Qt Creator, click **Open Project**, then navigate to "\examples\view_demo\PDFReader_Qt" folder, and choose **PDFReader_Qt.pro** file to open the demo.

   If the "**Configure Project"** window appears, please choose one kit according to the prompt, and then click **Configure Project** button as shown in Figure 2-5.



Configure Project

Qt Creator can use the following kits for project **PDFReader_Qt**:

The project **PDFReader_Qt** is not yet configured.
Qt Creator uses the kit **Desktop Qt 5.2.1 GCC 32bit** to parse the project.

✓ **Desktop Qt 5.2.1 GCC 32bit**

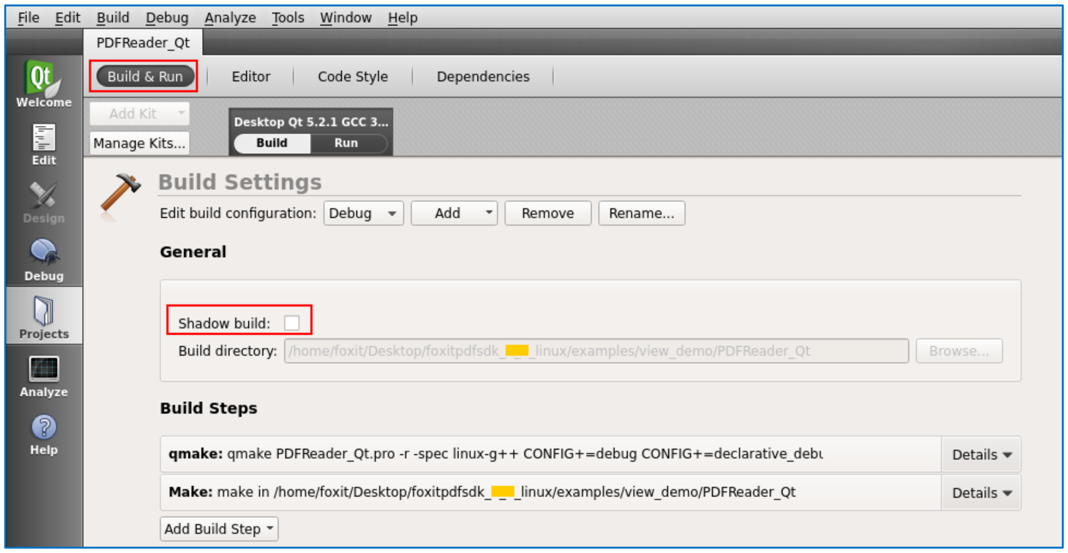
✓ Debug pdfsdk_[redacted]_linux/examples/view_demo/build-PDFReader_Qt-Desktop_Qt_5_2_1_GCC_32bit-Debug

✓ Release dfsdk_[redacted]_linux/examples/view_demo/build-PDFReader_Qt-Desktop_Qt_5_2_1_GCC_32bit-Release

Import Build From...

Figure 2-5

2) After opening the demo, click **Projects** on the left toolbar, uncheck the **Shadow build** option under "**Build&Run** -> **Build Settings** -> **General**" as shown in Figure 2-6.



Build Settings

Edit build configuration: Debug

General

Shadow build: ☐

Build directory: /home/foxit/Desktop/foxitpdfsdk_[redacted]_linux/examples/view_demo/PDFReader_Qt

Build Steps

qmake: qmake PDFReader_Qt.pro -r -spec linux-g++ CONFIG+=debug CONFIG+=declarative_debu

Make: make in /home/foxit/Desktop/foxitpdfsdk_[redacted]_linux/examples/view_demo/PDFReader_Qt

Figure 2-6

3) Compile and run the demo. Click **Build** -> **Run** to run the demo. After the demo starts, you will see the following window:

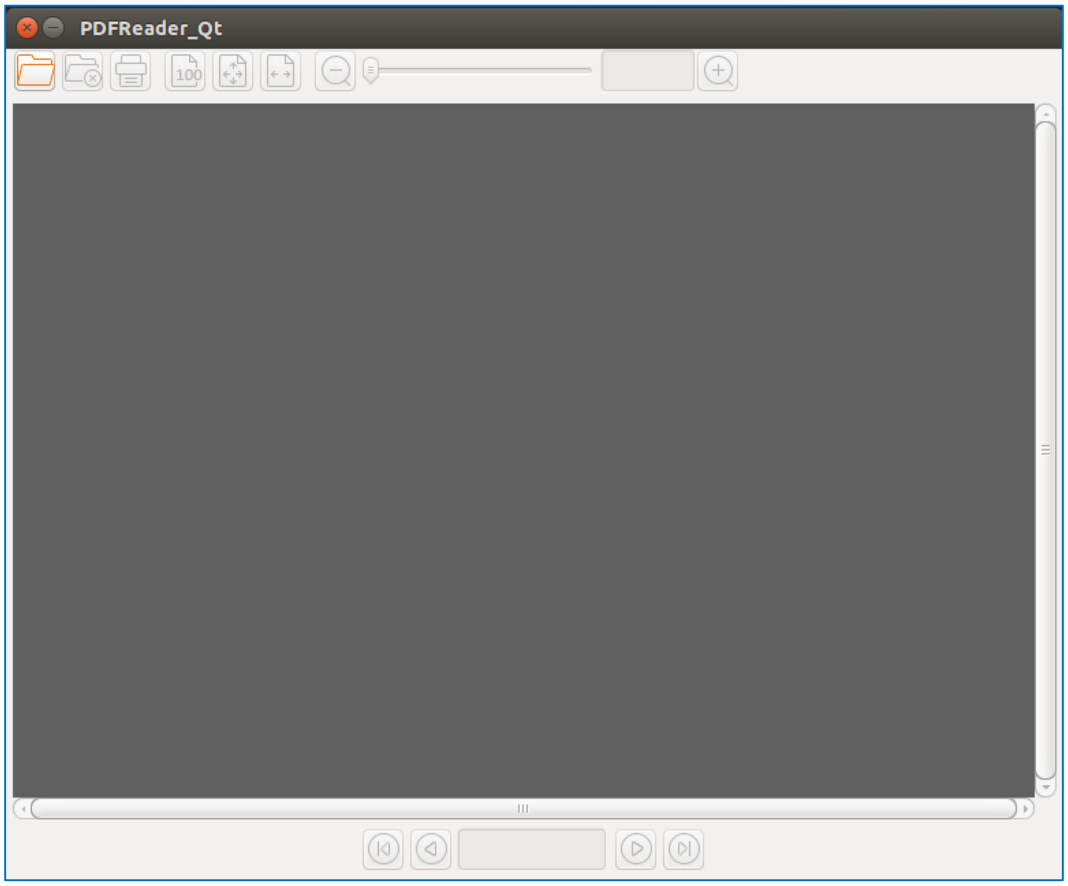
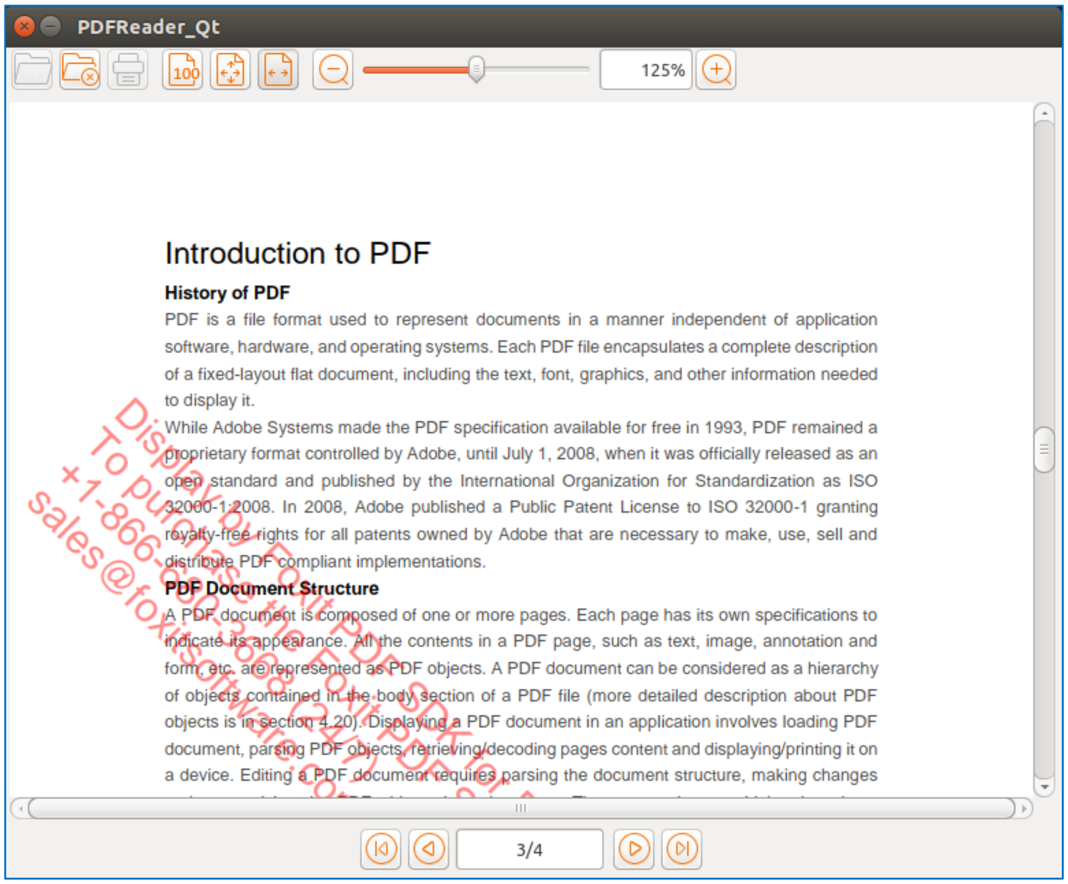


Figure 2-7

Click 📂 on the above image (Figure 2-7) to open a PDF document. Then you can browse the content by scrolling down, turn to the next page, and zoom in/out the PDF page as shown in Figure 2-8.



Introduction to PDF

History of PDF

PDF is a file format used to represent documents in a manner independent of application software, hardware, and operating systems. Each PDF file encapsulates a complete description of a fixed-layout flat document, including the text, font, graphics, and other information needed to display it.

While Adobe Systems made the PDF specification available for free in 1993, PDF remained a proprietary format controlled by Adobe, until July 1, 2008, when it was officially released as an open standard and published by the International Organization for Standardization as ISO 32000-1:2008. In 2008, Adobe published a Public Patent License to ISO 32000-1 granting royalty-free rights for all patents owned by Adobe that are necessary to make, use, sell and distribute PDF compliant implementations.

PDF Document Structure

A PDF document is composed of one or more pages. Each page has its own specifications to indicate its appearance. All the contents in a PDF page, such as text, image, annotation and form, etc. are represented as PDF objects. A PDF document can be considered as a hierarchy of objects contained in the body section of a PDF file (more detailed description about PDF objects is in section 4.20). Displaying a PDF document in an application involves loading PDF document, parsing PDF objects, retrieving/decoding pages content and displaying/printing it on a device. Editing a PDF document requires parsing the document structure, making changes

Figure 2-8

2.3.3 Create a simple project

In this section, we will show you how to use Foxit PDF SDK for Linux to create a simple project that renders the first page of a PDF to a bitmap and saves it as a JPG image. Please follow the steps below:

1) Create a folder called "test_linux".

2) Copy "include" and "lib" folders from "foxitpdfsdk_7_3_linux" folder to the project "test_linux" folder.

3) Create a "test_linux.cpp" file under "test_linux" folder, and add the code similar to the "test_win.cpp" described in section 2.2.3 "How to create a simple project".

   The "test_linux.cpp" will look like as follows: (For better viewing, I paste the code to Visual Studio to highlight the code)

```
#include <iostream>
#include "common/fs_common.h"
```

```
#include "pdf/fs_pdfdoc.h"
#include "pdf/fs_pdfpage.h"
#include "common/fs_render.h"

using namespace std;
using namespace foxit;
using namespace common;
using namespace pdf;
using namespace foxit::common;

int main(int argc, char* argv[])

{
        // The value of "sn" can be got from "gsdk_sn.txt" (the string after "SN=").
        // The value of "key" can be got from "gsdk_key.txt" (the string after "Sign=").
        const char* sn = " ";
        const char* key = " ";
        foxit::ErrorCode code = Library::Initialize(sn, key);
        if (code != foxit::e_ErrSuccess) {
                return FALSE;
        }

        // Load a PDF document, and parse the first page of the document.
        PDFDoc doc("../../Sample.pdf");
        ErrorCode error_code = doc.Load();
        if (error_code!= foxit::e_ErrSuccess) return 0;
        PDFPage page = doc.GetPage(0);
        page.StartParse(foxit::pdf::PDFPage::e_ParsePageNormal, NULL, false);

        int width = static_cast<int>(page.GetWidth());
        int height = static_cast<int>(page.GetHeight());
        Matrix matrix = page.GetDisplayMatrix(0, 0, width, height, page.GetRotation());

        // Prepare a bitmap for rendering.
        Bitmap bitmap(width, height, Bitmap::e_DIBArgb, NULL, 0);
        bitmap.FillRect(0xFFFFFFFF, NULL);
        // Render page.
        Renderer render(bitmap, false);
        render.StartRender(page, matrix, NULL);
        // Add the bitmap to image and save the image.
        Image img;
        img.AddFrame(bitmap);
        img.SaveAs("testpage.jpg");
        return 0;
}
```

4) Put a "Sample.pdf" document into the project "test_linux" folder.

5) Create a Makefile. In this Makefile, the PDF SDK library shall be included in the build path. Use libfsdk_linux64.so for 64 bit system or libfsdk_linux32.so for 32 bit system. A sample Makefile is as follows:

```
CXX=g++

# Foxit PDF SDK lib and head files include
INCLUDE_PATH=-Iinclude
FSDKLIB_PATH=-Llib
LIB_PATH=lib
FSDKLIB=-lfsdk_linux64
LIBNAME=libfsdk_linux64.so
LIBS=$(FSDKLIB) -lpthread
LDFLAGS=-Wl,--rpath=.
DEST_PATH=./bin/rel_gcc
OBJ_PATH=./obj/rel
CCFLAGS=-c

# Copy the given library to the destination path
CP_LIB=cp $(LIB_PATH)/$(LIBNAME) $(DEST_PATH)

DEST=-o $(DEST_PATH)/$@
OBJ_DEST= -o $(OBJ_PATH)/$@

all: test_linux

dir:
        mkdir -p $(DEST_PATH)
        mkdir -p $(OBJ_PATH)

test_linux.o :test_linux.cpp
        $(CXX) $(CCFLAGS) $(INCLUDE_PATH) $^ $(OBJ_DEST)
```

```
test_linux: dir test_linux.o
        $(CXX) $(OBJ_PATH)/test_linux.o $(DEST) $(FSDKLIB_PATH) $(LIBS) $(LDFLAGS)
```

6) Build the project. Open a terminal window, navigate to "test_linux", and run "**make test_linux**" to generate binary file in "test_linux\bin\rel_gcc" folder.

7) Execute the binary file. Navigate to the folder with the terminal, run "**./test_linux**", and the "testpage.jpg" will be generated in current folder.

2.4 Mac

In this guide, one thing to note is that the highlighted rectangle in the figure is the version of the SDK. Here the SDK version is 7.3, so it shows 7_3.

2.4.1 What is in this package

Download Foxit PDF SDK zip for Mac package and extract it to a new directory "foxitpdfsdk_7_3_mac". The structure of the release package is shown in Figure 2-9. This package contains the following folders:

doc:	API references, developer guide
examples:	sample projects and demos
include:	header files for Foxit PDF SDK API
lib:	libraries and license files

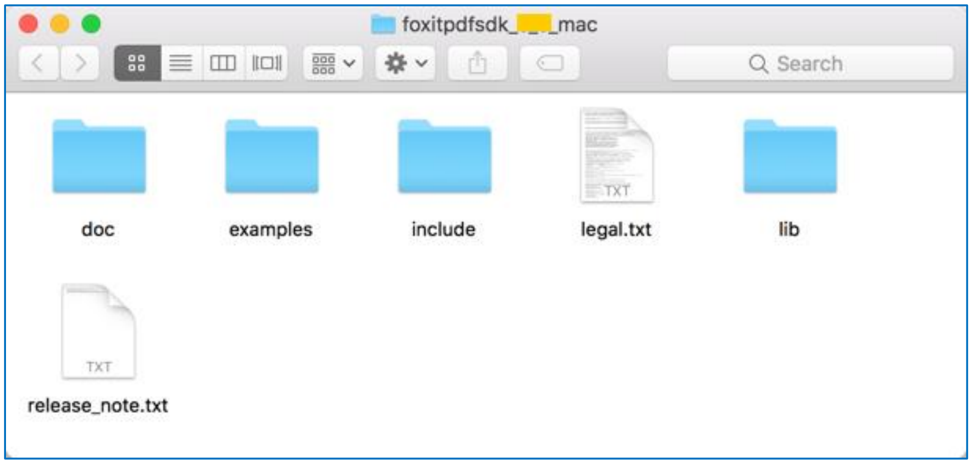


Figure 2-9

In "examples\simple_demo" folder, there are a wide range of PDF document application demos.

2.4.2 How to run a demo

Before running the demos, please make sure you have configured the environment correctly and installed CMake (3.1 or later) on your machine.

OS	Compiler tools or IDE	Version
Mac	Xcode	8 and later

To run a demo in a terminal window (except security, signature, compliance and html2pdf demos which will be introduced later), please follow the steps:

1) Open a terminal window, navigate to "foxitpdfsdk_7_3_mac\examples\simple_demo";
2) Run "**cmake -DPRJ_NAME=XXX**" to compile a specific demo. "XXX" is the demo name, which must be "annotation", "attachment", "pdf2text" and so on. For example, run "cmake -DPRJ_NAME=annotation".
3) Run "**make**" to build the demo specified in the above command. Then the executable file will be generated named "**XXX_xxx**". "XXX" is the demo name and "xxx" is the architecture name, such as "annotation_mac64".
4) Run "**./XXX_xxx**" to run the demo. For example, run "./annotation_mac64" to run the annotation demo.

"\examples\simple_demo\input_files" contains all the input files used among these demos. Some demos will generate output files (pdf, text or image files) to a folder named by the demo name under "examples/simple_demo/output_files/".

Security and Signature demos

Before running **security** and **signature** demos, please make sure that you have already installed OpenSSL. Download an OpenSSL source package from the OpenSSL website, or you can contact us directly, and then extract it and do the following:

1) Put the OpenSSL folder into the "include" folder which ensures that the OpenSSL header files included in the demos can be found.
2) Put the "libssl.a" and "libcrypto.a" libraries into the "lib" folder.
3) Run the demos following the steps as the other demos.

Note*: We have verified that OpenSSL 1.0.2m version is available on the security and signature demos, you can replaced it with the desired version, and maybe need to do some changes.*

Compliance demo

For how to run the **compliance** demo, please refer to section 3.35 "Compliance".

HTML to PDF demo

For how to run the **html2pdf** demo, please refer to section 3.37 "HTML to PDF Conversion".

2.4.3 How to create a simple project

In this section, we will show you how to use Foxit PDF SDK for Mac (C++) to create a simple project that renders the first page of a PDF to a bitmap and saves it as a JPG image. Please follow the steps below:

1) Create a folder called "test_mac".

2) Copy "include" and "lib" folders from "foxitpdfsdk_7_3_mac" folder to the project "test_mac" folder.

3) Create a "test_mac.cpp" file under "test_mac" folder, and add the code similar to the "test_win.cpp" described in section 2.2.3 "How to create a simple project".

   The "test_mac.cpp" will look like as follows:

```
#include <iostream>
#include "common/fs_common.h"
#include "pdf/fs_pdfdoc.h"
#include "pdf/fs_pdfpage.h"
#include "common/fs_render.h"

using namespace std;
using namespace foxit;
using namespace common;
using namespace pdf;
using namespace foxit::common;

int main(int argc, char* argv[])

{
  // The value of "sn" can be got from "gsdk_sn.txt" (the string after "SN=").
  // The value of "key" can be got from "gsdk_key.txt" (the string after "Sign=").
  const char* sn = " ";
  const char* key = " ";
```

```
    foxit::ErrorCode code = Library::Initialize(sn, key);
    if (code != foxit::e_ErrSuccess) {
      return FALSE;
    }

    // Load a PDF document, and parse the first page of the document.
    PDFDoc doc("../../Sample.pdf");
    ErrorCode error_code = doc.Load();
    if (error_code!= foxit::e_ErrSuccess) return 0;
    PDFPage page = doc.GetPage(0);
    page.StartParse(foxit::pdf::PDFPage::e_ParsePageNormal, NULL, false);

    int width = static_cast<int>(page.GetWidth());
    int height = static_cast<int>(page.GetHeight());
    Matrix matrix = page.GetDisplayMatrix(0, 0, width, height, page.GetRotation());

    // Prepare a bitmap for rendering.
    Bitmap bitmap(width, height, Bitmap::e_DIBArgb, NULL, 0);
    bitmap.FillRect(0xFFFFFFFF, NULL);
    // Render page.
    Renderer render(bitmap, false);
    render.StartRender(page, matrix, NULL);

    // Add the bitmap to image and save the image.
    Image img;
    img.AddFrame(bitmap);
    img.SaveAs("testpage.jpg");
    return 0;
}
```

4) Put a "Sample.pdf" document into the project "test_mac" folder.

5) Create a Makefile. In this Makefile, the PDF SDK library shall be included in the build path. Use libfsdk_mac64.dylib. A sample Makefile is as follows:

```
CXX=g++

# Foxit PDF SDK lib and head files include
INCLUDE_PATH=-Iinclude
LIBNAME=./lib/libfsdk_mac64.dylib
LDFLAGS=-Wl,-rpath,../../lib
```

```
DEST_PATH=./bin/rel_gcc
OBJ_PATH=./obj/rel
CCFLAGS=-c

DEST=-o $(DEST_PATH)/$@
OBJ_DEST= -o $(OBJ_PATH)/$@

all: test_mac

dir:
	mkdir -p $(DEST_PATH)
	mkdir -p $(OBJ_PATH)

test_mac.o :test_mac.cpp
	$(CXX) $(CCFLAGS) $(INCLUDE_PATH) $^ $(OBJ_DEST)

test_mac: dir test_mac.o
	$(CXX) $(OBJ_PATH)/test_mac.o $(DEST) $(LDFLAGS) $(LIBNAME)
```

6) Build the project. Open a terminal window, navigate to "test_mac", and run "make test_mac" to generate binary file in "test_mac\bin\rel_gcc" folder.

7) Execute the binary file. Navigate to the folder with the terminal, run "./test_mac", and the "testpage.jpg" will be generated in current folder.

3 WORKING WITH SDK API

In this section, we will introduce a set of major features and list some examples for each feature to show you how to integrate powerful PDF capabilities with your applications using Foxit PDF SDK C++ API. You can refer to the API reference [2] to get more details about the APIs used in all of the examples.

3.1 Initialize Library

It is necessary for applications to initialize Foxit PDF SDK before calling any APIs. The function foxit::common::Library::Initialize is provided to initialize Foxit PDF SDK. A license should be purchased for the application and pass unlock key and code to get proper supports. When there is no need to use Foxit PDF SDK any more, please call function foxit::common::Library::Release to release it.

Note *The parameter "sn" can be found in the "**gsdk_sn.txt**" (the string after "SN=") and the "key" can be found in the "**gsdk_key.txt**" (the string after "Sign=").*

Example:

3.1.1 How to initialize Foxit PDF SDK

```
#include "include/common/fs_common.h"

using namespace foxit;
using namespace common;
...

const char* sn = " ";
const char* key = " ";
foxit::ErrorCode code = Library::Initialize(sn, key);
if (code != foxit::e_ErrSuccess) {
            return FALSE;
}
```

3.2 Document

A PDF document object can be constructed with an existing PDF file from file path, memory buffer, a custom implemented ReaderCallback object and an input file stream. Then call function PDFDoc::Load or PDFDoc::StartLoad to load document content. A PDF document object is used for document level operation, such as opening and closing files, getting page, metadata and etc.

Example:

3.2.1 How to create a PDF document from scratch

```
#include "include/pdf/fs_pdfdoc.h"

using namespace foxit;
using namespace common;
using namespace pdf;
...

PDFDoc doc();
```

***Note**: It creates a new PDF document without any pages.*

3.2.2 How to load an existing PDF document from file path

```
#include "include/pdf/fs_pdfdoc.h"

using namespace foxit;
using namespace common;
using namespace pdf;
...

PDFDoc doc("Sample.pdf");
ErrorCode error_code = doc.Load();
if (error_code!= foxit::e_ErrSuccess) return 0;
```

3.2.3 How to load an existing PDF document from a memory buffer

```
#include "include/pdf/fs_pdfdoc.h"

using namespace foxit;
using namespace common;
using namespace pdf;
...

FILE* pFile = fopen(TEST_DOC_PATH"blank.pdf", "rb");
ASSERT_EQ(TRUE, NULL != pFile);
fseek(pFile, 0, SEEK_END);
long lFileSize = ftell(pFile);
char* buffer = new char[lFileSize];
memset(buffer, 0, sizeof(char)*lFileSize);
fseek(pFile, 0, SEEK_SET);
fread(buffer, sizeof(char), lFileSize, pFile);
fclose(pFile);
```

```
PDFDoc doc = PDFDoc(buffer, lFileSize);
ErrorCode error_code = doc.Load();
if (error_code!= foxit::e_ErrSuccess) return 0;
```

3.2.4 How to load an existing PDF document from a file read callback object

```
#include "include/pdf/fs_pdfdoc.h"


using namespace foxit;
using namespace common;
using namespace pdf;
...

class CFSFile_Read : public ReaderCallback
{
public:
        CFSFile_Read():m_fileFP(NULL)
                ,m_bLargeFile(FALSE)
        {}

        ~CFSFile_Read() {}

        bool LoadFile(const wchar_t* wFilePath, bool bLargeFile = FALSE)
        {
                std::wstring strTemp(wFilePath);
                string bstrFilepath = wchar2utf8(strTemp.c_str(), strTemp.size());

                m_fileFP = fopen(bstrFilepath.c_str(), "rb");
                if (!m_fileFP) return FALSE;

                m_bLargeFile = bLargeFile;
                return TRUE;
        }

        bool LoadFile(const char* filePath, bool bLargeFile = FALSE)
        {
                m_fileFP = fopen(filePath, "rb");
                if (!m_fileFP) return FALSE;

                m_bLargeFile = bLargeFile;
                return TRUE;
        }

        FILESIZE GetSize()
        {
                if (m_bLargeFile)
                {
#if defined(_WIN32) || defined(_WIN64)
                        _fseeki64(m_fileFP, 0, SEEK_END);
                        long long sizeL = _ftelli64(m_fileFP);
#elif defined(__linux__) || defined(__APPLE__)
```

```
                        fseeko(m_fileFP, 0, SEEK_END);
                        long long sizeL = ftello(m_fileFP);
#endif
                        return sizeL;
                }
                else
                {
                        fseek(m_fileFP, 0, SEEK_END);
                        return (uint32)ftell(m_fileFP);
                }
        }

        int ReadBlock(void* buffer, FILESIZE offset, size_t size)
        {
                if (m_bLargeFile)
                {
#if defined(_WIN32) || defined(_WIN64)
                        _fseeki64(m_fileFP, offset, SEEK_SET);
#elif defined(__linux__) || defined(__APPLE__)
                        fseeko(m_fileFP, offset, SEEK_SET);
#endif
                        long long readSize = fread(buffer, 1, size, m_fileFP);
                        return (readSize == size);
                }
                else
                {
                        if (!m_fileFP) return false;
                        if(0 != fseek(m_fileFP, offset, 0))
                                return false;
                        if(0 == fread(buffer, size, 1, m_fileFP))
                                return false;
                        return true;
                }
        }

        size_t  ReadBlock(void* buffer, size_t size) {
                if (m_bLargeFile)
                {
#if defined(_WIN32) || defined(_WIN64)
                        _fseeki64(m_fileFP, 0, SEEK_SET);
#elif defined(__linux__) || defined(__APPLE__)
                        fseeko(m_fileFP, 0, SEEK_SET);
#endif
                        return fread(buffer, 1, size, m_fileFP);
                }
                else
                {
                        if (!m_fileFP) return false;
                        if(0 != fseek(m_fileFP, 0, 0))
                                return 0;
                        return fread(buffer, size, 1, m_fileFP);
                }
        }

        void Release()
        {
                if(m_fileFP)
                        fclose(m_fileFP);
                m_fileFP = NULL;
                delete this;
        }

private:
        FILE* m_fileFP;
        bool m_bLargeFile;
}
...
string inputPDFPath = "Sample.pdf";
CFSFile_Read* pFileRead = new CFSFile_Read();
If(!pFileRead->LoadFile(inputPDFPath.c_str()))
        Return;
PDFDoc doc = PDFDoc(pFileRead);
ErrorCode error_code = doc.Load();
if (error_code!= foxit::e_ErrSuccess) return 0;
```

3.2.5 How to load PDF document and get the first page of the PDF document

```
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"

using namespace foxit;
using namespace common;
using namespace pdf;
...

PDFDoc doc("Sample.pdf");
ErrorCode error_code = doc.Load();
if (error_code!= foxit::e_ErrSuccess) return 0;
PDFPage page = doc.GetPage(0);
page.StartParse(foxit::pdf::PDFPage::e_ParsePageNormal, NULL, false);
```

3.2.6 How to save a PDF to a file

```
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"

using namespace foxit;
using namespace common;
using namespace pdf;
```

```
…

PDFDoc doc("Sample.pdf");
ErrorCode error_code = doc.Load();
if (error_code!= foxit::e_ErrSuccess) return 0;
doc.SaveAs("new_Sample.pdf", PDFDoc::e_SaveFlagNoOriginal);
```

3.2.7 How to save a document into memory buffer by WriterCallback

```
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"


using namespace foxit;
using namespace common;
using namespace pdf;
…

// FileWriter for saving file to memory buffer.
class  FileWriter:public common::file::WriterCallback
{
public:
        FileWriter() {}

        ~FileWriter()  {}

        FILESIZE GetSize() {
                return  binary_buffer_.GetSize();
        }

        FX_BOOL Flush() {
                return  TRUE;
        }

        FX_BOOL WriteBlock(const void* buffer,FILESIZE offset,size_t size) {
                return  binary_buffer_.InsertBlock(offset,buffer,size);
        }

        FX_BOOL  ReadBlock(void* buffer,FILESIZE offset,size_t size)    {
                FX_LPBYTE byte_buffer = binary_buffer_.GetBuffer();
                memcpy(buffer, byte_buffer + offset, size);
        }

        void Release() {
        }

        CFX_BinaryBuf GetBuffer() {
                return binary_buffer_;
        }
private:
        CFX_BinaryBuf binary_buffer_;
};
...

FileWriter* filewriter = new FileWriter();

// Assuming PDFDoc doc has been loaded.
...

doc.StartSaveAs(filewriter, PDFDoc::e_SaveFlagNoOriginal);
...
```

3.3 Page

PDF Page is the basic and important component of PDF Document. A foxit::pdf::PDFPage object is retrieved from a PDF document by function foxit::pdf::PDFDoc::GetPage. Page level APIs provide functions to parse, render, edit (includes creating, deleting, flattening and etc.) a page, retrieve PDF annotations, read and set the properties of a page, and etc. For most cases, A PDF page needs to be parsed before it is rendered or processed.

Example:

3.3.1 How to get page size

```
#include "include/pdf/fs_pdfpage.h"


using namespace foxit;
using namespace common;
using namespace pdf;
...
// Assuming PDFPage page has been loaded and parsed.

int width = static_cast<int>(page.GetWidth());
int height = static_cast<int>(page.GetHeight());
```

3.3.2 How to calculate bounding box of page contents

```
#include "include/pdf/fs_pdfpage.h"


using namespace foxit;
using namespace common;
using namespace pdf;
...
//Assuming PDFDoc doc has been loaded.
//Assuming PDFPage page has been loaded and parsed.

RectF ret = page.CalcContentBBox(PDFPage::e_CalcContentsBox);
```

```
...
```

3.3.3 How to create a PDF page and set the size

```
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"

using namespace foxit;
using namespace common;
using namespace pdf;
...
// Assuming PDFDoc doc has been loaded.

PDFPage page = doc.InsertPage(index, PageWidth, PageHeight);
```

3.3.4 How to delete a PDF page

```
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"

using namespace foxit;
using namespace common;
using namespace pdf;
...
// Assuming PDFDoc doc has been loaded.

// Remove a PDF page by page index.
doc.RemovePage(index);

// Remove a specified PDF page.
doc.RemovePage(&page);
...
```

3.3.5 How to flatten a PDF page

```
#include "include/pdf/fs_pdfpage.h"

using namespace foxit;
using namespace common;
using namespace pdf;
...
// Assuming PDFPage page has been loaded and parsed.

// Flatten all contents of a PDF page.
page.Flatten(true, PDFPage::e_FlattenAll);
```

```
// Flatten a PDF page without annotations.
page.Flatten(true, PDFPage::e_FlattenNoAnnot);
```

```
// Flatten a PDF page without form controls.
page.Flatten(true, PDFPage::e_FlattenNoFormControl);

// Flatten a PDF page without annotations and form controls (Equals to nothing to be flattened).
page.Flatten(true, PDFPage::e_FlattenNoAnnot | PDFPage::e_FlattenNoFormControl);
...
```

3.3.6 How to get and set page thumbnails in a PDF document

```
#include "include/pdf/fs_pdfpage.h"


using namespace foxit;
using namespace common;
using namespace pdf;
...
// Assuming PDFPage page has been loaded and parsed.

Bitmap bmp();
// Write bitmap data to the bmp object.
...
// Set thumbnails to the page.
page.SetThumbnail(bmp);
// Load thumbnails in the page.
Bitmap bitmap = page.LoadThumbnail();
...
```

3.4 Render

PDF rendering is realized through the Foxit renderer, a graphic engine that is used to render page to a bitmap or platform graphics device. Foxit PDF SDK provides APIs to set rendering options/flags, for example set flag to decide whether to render form fields and signature, whether to draw image anti-aliasing and path anti-aliasing. To do rendering, you can use the following APIs:

- To render page and annotations, first use function Renderer::SetRenderContentFlags to decide whether to render page and annotation both or not, and then use function Renderer::StartRender to do the rendering. Function Renderer::StartQuickRender can also be used to render page but only for thumbnail purpose.
- To render a single annotation, use function Renderer::RenderAnnot.
- To render on a bitmap, use function Renderer::StartRenderBitmap.
- To render a reflowed page, use function Renderer::StartRenderReflowPage.

Widget annotation is always associated with form field and form control in Foxit PDF SDK. For how to render widget annotations, here is a recommended flow:

- After loading a PDF page, first render the page and all annotations in this page (including widget annotations).
- Then, if use pdf::interform::Filler object to fill the form, the function pdf::interform::Filler::Render should be used to render the focused form control instead of the function Renderer::RenderAnnot.

Example:

3.4.1 How to render a page to a bitmap

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"
#include "include/common/fs_render.h"

using namespace foxit;
using namespace common;
using namespace pdf;
using namespace foxit::common;

// Assuming PDFPage page has been loaded and parsed.

int width = static_cast<int>(page.GetWidth());
int height = static_cast<int>(page.GetHeight());
Matrix matrix = page.GetDisplayMatrix(0, 0, width, height, page.GetRotation());

// Prepare a bitmap for rendering.
Bitmap bitmap(width, height, Bitmap::e_DIBArgb, NULL, 0);
bitmap.FillRect(0xFFFFFFFF, NULL);
// Render page.
Renderer render(bitmap, false);
render.StartRender(page, matrix, NULL);
...
```

3.4.2 How to render page and annotation

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"
#include "include/common/fs_render.h"

using namespace foxit;
using namespace common;
using namespace pdf;
using namespace foxit::common;

// Assuming PDFPage page has been loaded and parsed.
int width = static_cast<int>(page.GetWidth());
int height = static_cast<int>(page.GetHeight());
Matrix matrix = page.GetDisplayMatrix(0, 0, width, height, page.GetRotation());

// Prepare a bitmap for rendering.
Bitmap bitmap(width, height, Bitmap::e_DIBArgb, NULL, 0);
bitmap.FillRect(0xFFFFFFFF, NULL);

Renderer render(bitmap, false);
uint32 dwRenderFlag = Renderer::e_RenderAnnot | Renderer::e_RenderPage;
render.SetRenderContentFlags(dwRenderFlag);
render.StartRender(page, matrix, NULL);
...
```

3.5 Attachment

In Foxit PDF SDK, attachments are only referred to attachments of documents rather than file attachment annotation, which allow whole files to be encapsulated in a document, much like email attachments. PDF SDK provides applications APIs to access attachments such as loading attachments, getting attachments, inserting/removing attachments, and accessing properties of attachments.

Example:

3.5.1 How to export the embedded attachment file from a PDF and save it as a single file

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfattachments.h"


using namespace foxit;
using namespace common;
using namespace pdf;
using namespace foxit::common;

// Assuming PDFDoc doc has been loaded.

// Get information of attachments.
Attachments attachments(doc);
int count = attachments.GetCount();
for (int i = 0; i < count; i++) {
   WString key = attachments.GetKey(i);
   FileSpec file_spec = attachments.GetEmbeddedFile(key);
```

```
    if (!file_spec.IsEmpty()) {
              WString name = file_spec.GetFileName();
              if (file_spec.IsEmbedded()) {
                 WString exFilePath = "output_directory";
                 file_spec.ExportToFile(exFilePath);
              }
    }
}
...
```

3.5.2 How to remove all the attachments of a PDF

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfattachments.h"
...
using namespace foxit;
using namespace common;
using namespace pdf;
using namespace foxit::common;

// Assuming PDFDoc doc has been loaded.

// Get information of attachments.
Attachments attachments(doc);
int count = attachments.GetCount();
for (int i = 0; i < count; i++) {
    WString key = attachments.GetKey(i);
    attachment.RemoveEmbeddedFile(&key);
}
...
```

3.6 Text Page

Foxit PDF SDK provides APIs to extract, select, search and retrieve text in PDF documents. PDF text contents are stored in TextPage objects which are related to a specific page. TextPage class can be used to retrieve information about text in a PDF page, such as single character, single word, text content within specified character range or rectangle and so on. It also can be used to construct objects of other text related classes to do more operations for text contents or access specified information from text contents:

- To search text in text contents of a PDF page, construct a TextSearch object with TextPage object.
- To access text such like hypertext link, construct a PageTextLinks object with TextPage object.

Example:

3.6.1 How to extract text from a PDF page

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_search.h"

using namespace std;
using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;
...

// Assuming PDFPage page has been loaded and parsed.

// Get the text page object.
TextPage text_page(page);
int count = text_page.GetCharCount();
if (count > 0) {
   WString text = text_page.GetChars();
   String s_text = text.UTF8Encode();
   fwrite((const char*)s_text, sizeof(char), s_text.GetLength(), file);
}
...
```

3.6.2 How to get the text within a rectangle area in a PDF

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_search.h"

using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;
...

RectF rect;
rect.left = 90;
rect.right = 450;
rect.top = 595;
rect.bottom = 580;
TextPage textPage = new TextPage (&page, TextPage::e_ParseTextNormal);
textPage.GetTextInRect(&rect);
...
```

3.7 Text Search

Foxit PDF SDK provides APIs to search text in a PDF document, a XFA document, a text page or in a PDF annotation's appearance. It offers functions to do a text search and get the searching result:

- To specify the searching pattern and options, use functions TextSearch::SetPattern, TextSearch::SetStartPage (only useful for a text search in PDF document), TextSearch::SetEndPage (only useful for a text search in PDF document) and TextSearch::SetSearchFlags.
- To do the searching, use function TextSearch::FindNext or TextSearch::FindPrev.
- To get the searching result, use function TextSearch::GetMatchXXX().

Example:

3.7.1 How to search a text pattern in a PDF

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"
#include "include/pdf/fs_search.h"

using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;
...

// Assuming PDFDoc doc has been loaded.

// Search for all pages of doc.
TextSearch search(doc, NULL);

int start_index = 0, end_index = doc.GetPageCount() - 1;
search.SetStartPage(start_index);
search.SetEndPage(end_index);

WString pattern = L"Foxit";
search.SetPattern(pattern);

foxit::uint32 flags = TextSearch::e_SearchNormal;
search.SetSearchFlags(flags);
...

int match_count = 0;
while (search.FindNext()) {
    RectFArray rect_array = search.GetMatchRects();
    match_count ++;
  }
...
```

3.8 Text Link

In a PDF page, some text contents that represent a hypertext link to a website or a resource on the intent, or an email address are the same with common texts. Prior to text link processing, user should first call PageTextLinks::GetTextLink to get a textlink object.

Example:

3.8.1 How to retrieve hyperlinks in a PDF page

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"
#include "include/pdf/fs_search.h"

using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;
...

// Assuming PDFPage page has been loaded and parsed.

// Get the text page object.
TextPage text_page(page);
PageTextLinks pageTextLink(text_page);
TextLink textLink = pageTextLink.GetTextLink(index);
String strURL = textLink.GetURI();
...
```

3.9 Bookmark

Foxit PDF SDK provides navigational tools called Bookmarks to allow users to quickly locate and link their point of interest within a PDF document. PDF bookmark is also called outline, and each bookmark contains a destination or actions to describe where it links to. It is a tree-structured hierarchy, so function pdf::PDFDoc::GetRootBookmark must be called first to get the root of the whole bookmark tree before accessing to the bookmark tree. Here, "root bookmark" is an abstract object which can only have some child bookmarks without next sibling bookmarks and any data (includes bookmark data, destination data and action data). It cannot be shown on the application UI since it has no data. Therefore, a root bookmark can only call function Bookmark::GetFirstChild.

After the root bookmark is retrieved, following functions can be called to access other bookmarks:

- To access the parent bookmark, use function Bookmark::GetParent.
- To access the first child bookmark, use function Bookmark::GetFirstChild.

- To access the next sibling bookmark, use function Bookmark::GetNextSibling.
- To insert a new bookmark, use function Bookmark::Insert.
- To move a bookmark, use function Bookmark::MoveTo.

Example:

3.9.1 How to find and list all bookmarks of a PDF

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_filespec.h"
#include "include/pdf/fs_bookmark.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"

using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;
...

// Assuming PDFDoc doc has been loaded.

Bookmark root = doc.GetRootBookmark();
Bookmark first_bookmark = root.GetFirstChild();

if (first_bookmark != null)
{
        TraverseBookmark(first_bookmark, 0);
}

Private void TraverseBookmark(Bookmark root, int iLevel)
{
        if (root != null)
        {
       Bookmark child = root.GetFirstChild();
            while (child != null)
            {
              TraverseBookmark(child, iLevel + 1);
              child = child.GetNextSibling();
            }
    }
}
...
```

3.9.2 How to insert a new bookmark

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_filespec.h"
#include "include/pdf/fs_bookmark.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"
```

```
using namespace std;
using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;

// Assuming PDFDoc doc has been loaded.

Bookmark root = doc.GetRootBookmark();
if (root.IsEmpty())
{
  root = doc.CreateRootBookmark();
}
Destination dest = Destination::CreateFitPage(doc, 0);
CFX_WideString ws_title;
ws_title.Format((FX_LPCWSTR)L"A bookmark to a page (index: %d)", 0);
Bookmark child = root.Insert(ws_title, foxit::pdf::Bookmark::e_PosLastChild);
child.SetDestination(dest);
child.SetColor(0xF68C21);
```

3.10 Form (AcroForm)

PDF currently supports two different forms for gathering information interactively from the user - AcroForms and XFA forms. Acroforms are the original PDF-based fillable forms, based on the PDF architecture. Foxit PDF SDK provides APIs to view and edit form field programmatically. Form fields are commonly used in PDF documents to gather data. The Form class offers functions to retrieve form fields or form controls, import/export form data and other features, for example:

- To retrieve form fields, please use functions Form::GetFieldCount and Form::GetField.
- To retrieve form controls from a PDF page, please use functions Form::GetControlCount and Form::GetControl.
- To import form data from an XML file, please use function Form::ImportFromXML; to export form data to an XML file, please use function Form::ExportToXML.
- To retrieve form filler object, please use function Form::GetFormFiller.

To import form data from a FDF/XFDF file or export such data to a FDF/XFDF file, please refer to functions pdf::PDFDoc::ImportFromFDF and pdf::PDFDoc::ExportToFDF.

Example:

3.10.1 How to load the forms in a PDF

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfform.h"
```

```
using namespace foxit;
using namespace pdf;
using namespace interform;
...

// Assuming PDFDoc doc has been loaded.

bool hasForm = doc.HasForm();
if(hasForm)
  Form form(doc);
...
```

3.10.2 How to count form fields and get/set the properties

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/interform/fs_pdfform.h"


using namespace foxit;
using namespace pdf;
using namespace interform;


...
// Assuming PDFDoc doc has been loaded.

Form form(doc);
int countFields = form. GetFieldCount(NULL);
for (int i = 0; i < nFieldCount; i++)
{
    Field field = form.GetField(i, filter);
    Field::Type type = field.GetType();
    WString org_alternateName = field.GetAlternateName();
    field.SetAlternateName(L"signature");
}
```

3.10.3 How to export the form data in a PDF to a XML file

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/interform/fs_pdfform.h"


using namespace foxit;
using namespace pdf;
using namespace interform;
...
// Assuming PDFDoc doc has been loaded.
```

```
Form form(doc);
...
form.ExportToXML(XMLFilePath);
...
```

3.10.4 How to import form data from a XML file

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/interform/fs_pdfform.h"

using namespace foxit;
using namespace pdf;
using namespace interform;
...
// Assuming PDFDoc doc has been loaded.

Form form(doc);
...
form.ImportFromXML(XMLFilePath);
...
```

3.11 XFA Form

XFA (XML Forms Architecture) forms are XML-based forms, wrapped inside a PDF. The XML Forms Architecture provides a template-based grammar and a set of processing rules that allow uses to build interactive forms. At its simplest, a template-based grammar defines fields in which a user provides data.

Foxit PDF SDK provides APIs to render the XFA form, fill the form, export or import form's data.

Note:

- *Foxit PDF SDK provides two callback classes foxit::addon::xfa::AppProviderCallback and foxit::addon::xfa::DocProviderCallback to represent the callback objects as an XFA document provider and an XFA application provider respectively. All the functions in those classes are used as callback functions. Pure virtual functions should be implemented by users.*
- *To use the XFA form feature, please make sure the license key has the permission of the 'XFA' module.*

Example:

3.11.1 How to load XFADoc and represent an Interactive XFA form

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"

#include "include/pdf/interform/fs_pdfform.h"
#include "include/addon/xfa/fs_xfa.h"

using namespace foxit;
using namespace foxit::common;

using namespace pdf;
using namespace foxit::addon::xfa;

CFS_XFAAppHandler* pXFAAppHandler = new CFS_XFAAppHandler(); // implement from
foxit::addon::xfa::AppProviderCallback
Library::RegisterXFAAppProviderCallback(pXFAAppHandler);
WString input_file = input_path + L"xfa_dynamic.pdf";
PDFDoc doc(input_file);
ErrorCode error_code = doc.Load();
if (error_code != foxit::e_ErrSuccess) {
	return 1;
}

CFS_XFADocHandler* pXFADocHandler = new CFS_XFADocHandler(); // implement from
foxit::addon::xfa::DocProviderCallback
XFADoc xfa_doc(doc, pXFADocHandler);
xfa_doc.StartLoad(NULL);
...
```

3.11.2 How to export and import XFA form data

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/interform/fs_pdfform.h"
#include "include/addon/xfa/fs_xfa.h"

using namespace foxit;
using namespace foxit::common;
using namespace pdf;
using namespace foxit::addon::xfa;

// Assuming FSXFADoc xfa_doc has been loaded.

xfa_doc.ExportData(L"xfa_form.xml", XFADoc::e_ExportDataTypeXML);

xfa_doc.ResetForm();
doc.SaveAs( L"xfa_dynamic_resetform.pdf");

xfa_doc.ImportData(L"xfa_form.xml");
doc.SaveAs(L"xfa_dynamic_importdata.pdf");
...
```

3.12 Form Filler

Form filler is the most commonly used feature for users. Form filler allows applications to fill forms dynamically. The key point for applications to fill forms is to construct some callback functions for PDF SDK to call. To fill the form, please construct a Filler object by current Form object or retrieve the Filler object by function Form::GetFormFiller if such object has been constructed. (There should be only one form filler object for an interactive form). For how to fill a form with form filler, you can refer to the view demo "**view_demo**" in the "\examples\view_demo" folder of the download package for Windows.

3.13 Form Design

Fillable PDF forms (AcroForm) are especially convenient for preparation of various applications, such as taxes and other government forms. Form design provides APIs to add or remove form fields (Acroform) to or from a PDF file. Designing a form from scratch allows developers to create the exact content and layout of the form they want.

Example:

3.13.1 How to add a text form field to a PDF

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"
#include "include/pdf/interform/fs_pdfform.h"

using namespace foxit;
using namespace pdf;
using namespace interform;
...
// Assuming PDFDoc doc has been loaded.
// Assuming PDFPage page has been loaded and parsed.

// Add text field
Control control = form.AddControl(page, L"Text Field0", Field::e_TypeTextField, RectF(50, 600, 90, 640));
control.GetField().SetValue(L"3");
// Update text field's appearance.
control.GetWidget().ResetAppearanceStream();

Control control1 = form.AddControl(page, L"Text Field1", Field::e_TypeTextField, RectF(100, 600, 140, 640));
control1.GetField().SetValue(L"123");
// Update text field's appearance.
control1.GetWidget().ResetAppearanceStream();
...
```

3.13.2 How to remove a text form field from a PDF

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"
#include "include/pdf/interform/fs_pdfform.h"

using namespace foxit;
using namespace pdf;
using namespace interform;
...
// Assuming PDFDoc doc has been loaded.

Form form(doc);
const wchar_t* filter = L"text1";
int countFields = form.GetFieldCount(NULL);
for (int i = 0; i < countFields; i++)
{
    Field field = form.GetField(i, filter);
    if (field.GetType() == Field::e_TypeTextField) {
            form.RemoveField(field);
    }
}
...
```

3.14 Annotations

3.14.1 General

An annotation associates an object such as note, line, and highlight with a location on a page of a PDF document. It provides a way to interact with users by means of the mouse and keyboard. PDF includes a wide variety of standard annotation types as listed in Table 3-1. Among these annotation types, many of them are defined as markup annotations for they are used primarily to mark up PDF documents. These annotations have text that appears as part of the annotation and may be displayed in other ways by a conforming reader, such as in a Comments pane. The 'Markup' column in Table 3-1 shows whether an annotation is a markup annotation.

Foxit PDF SDK supports most annotation types defined in PDF reference [1]. PDF SDK provides APIs of annotation creation, properties access and modification, appearance setting and drawing.

Table 3-1

Annotation type	Description	Markup	Supported by SDK
Text(Note)	Text annotation	Yes	Yes

Link	Link Annotation	No	Yes
FreeText (TypeWritter/TextBox/Callout)	Free text annotation	Yes	Yes
Line	Line annotation	Yes	Yes
Square	Square annotation	Yes	Yes
Circle	Circle annotation	Yes	Yes
Polygon	Polygon annotation	Yes	Yes
PolyLine	PolyLine annotation	Yes	Yes
Highlight	Highlight annotation	Yes	Yes
Underline	Underline annotation	Yes	Yes
Squiggly	Squiggly annotation	Yes	Yes
StrikeOut	StrikeOut annotation	Yes	Yes
Stamp	Stamp annotation	Yes	Yes
Caret	Caret annotation	Yes	Yes
Ink(pencil)	Ink annotation	Yes	Yes
Popup	Popup annotation	No	Yes
File Attachment	FileAttachment annotation	Yes	Yes
Sound	Sound annotation	Yes	No
Movie	Movie annotation	No	No
Widget*	Widget annotation	No	Yes
Screen	Screen annotation	No	Yes
PrinterMark	PrinterMark annotation	No	No
TrapNet	Trap network annotation	No	No
Watermark*	Watermark annotation	No	Yes
3D	3D annotation	No	No
Redact	Redact annotation	Yes	Yes

Note:

1. The annotation types of widget and watermark are special. They aren't supported in the module of 'Annotation'. The type of widget is only used in the module of 'form filler' and the type of watermark only in the module of 'watermark'.
2. Foxit SDK supports a customized annotation type called PSI (pressure sensitive ink) annotation that is not described in PDF reference [1]. Usually, PSI is for handwriting features and Foxit SDK treats it as PSI annotation so that it can be handled by other PDF products.

Example:

3.14.1.1 *How to add a link annotation to a PDF page*

```
#include "include/common/fs_common.h"
#include "include/pdf/actions/fs_action.h"
#include "include/pdf/annots/fs_annot.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/objects/fs_pdfobject.h"
#include "include/pdf/fs_pdfpage.h"

using namespace foxit;
using namespace foxit::common;
using namespace pdf;
using namespace annots;

// Assuming PDFPage page has been loaded and parsed.
// Assuming the annnots in the page have been loaded.

// Add link annotation.
annots::Link link(page.AddAnnot( Annot::e_Link, RectF(350,350,380,400)));
link.SetHighlightingMode(Annot::e_HighlightingToggle);
...
```

3.14.1.2 *How to add a highlight annotation to a page and set the related annotation properties*

```
#include "include/common/fs_common.h"
#include "include/pdf/actions/fs_action.h"
#include "include/pdf/annots/fs_annot.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/objects/fs_pdfobject.h"
#include "include/pdf/fs_pdfpage.h"

using namespace foxit;
using namespace foxit::common;
using namespace pdf;
using namespace annots;

// Assuming PDFPage page has been loaded and parsed.
// Assuming the annnots in the page have been loaded.

// Add highlight annotation.
annots::Highlight highlight(page.AddAnnot(Annot::e_Highlight,RectF(10,450,100,550)));
highlight.SetContent(L"Highlight");
annots::QuadPoints quad_points;
quad_points.first = PointF(10, 500);
quad_points.second = PointF(90, 500);
quad_points.third = PointF(10, 480);
quad_points.fourth = PointF(90, 480);
annots::QuadPointsArray quad_points_array;
quad_points_array.Add(quad_points);
highlight.SetQuadPoints(quad_points_array);
highlight.SetSubject(L"Highlight");
highlight.SetTitle(L"Foxit SDK");
highlight.SetCreationDateTime(GetLocalDateTime());
highlight.SetModifiedDateTime(GetLocalDateTime());
highlight.SetUniqueID(WString::FromLocal(RandomUID()));

// Appearance should be reset.
highlight.ResetAppearanceStream();
...
```

3.14.1.3 How to set the popup information when creating markup annotations

```
#include "include/common/fs_common.h"
#include "include/pdf/actions/fs_action.h"
#include "include/pdf/annots/fs_annot.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/objects/fs_pdfobject.h"
#include "include/pdf/fs_pdfpage.h"


using namespace foxit;
using namespace foxit::common;
using namespace pdf;
using namespace annots;

// Assuming PDFPage page has been loaded and parsed.
// Assuming the annnots in the page have been loaded.

// Create a new note annot and set the properties for it.
annots::Note note(page.AddAnnot(Annot::e_Note, RectF(10,350,50,400)));
note.SetIconName("Comment");
note.SetSubject(L"Note");
note.SetTitle(L"Foxit SDK");
note.SetContent(L"Note annotation.");
note.SetCreationDateTime(GetLocalDateTime());
note.SetModifiedDateTime(GetLocalDateTime());
note.SetUniqueID(WString::FromLocal(RandomUID()));

// Create a new popup annot and set it to the new note annot.
Popup popup(page.AddAnnot(Annot::e_Popup, RectF(300,450,500,550)));
popup.SetBorderColor(0x00FF00);
popup.SetOpenStatus(false);
popup.SetModifiedDateTime(GetLocalDateTime());
note.SetPopup(popup);
...
```

3.14.1.4 *How to get a specific annotation in a PDF using device coordinates*

```
#include "include/common/fs_common.h"
#include "include/pdf/actions/fs_action.h"
#include "include/pdf/annots/fs_annot.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/objects/fs_pdfobject.h"
#include "include/pdf/fs_pdfpage.h"

using namespace foxit;
using namespace foxit::common;
using namespace pdf;
using namespace annots;

// Assuming PDFDoc doc has been loaded.
// Assuming PDFPage page has been loaded and parsed.
...

int width = static_cast<int>(page.GetWidth());
int height = static_cast<int>(page.GetHeight());

// Get page transformation matrix.
Matrix displayMatrix= page.GetDisplayMatrix(0, 0, width, height, page.GetRotation());
int iAnnotCount = page.GetAnnotCount();

for(int i=0; i<iAnnotCount; i++)
{
        Annot pAnnot = page.GetAnnot(i);
        ASSERT_FALSE(pAnnot.IsEmpty());
        if (Annot::e_Popup == pAnnot.GetType()) continue;
        RectI annotRect = pAnnot.GetDeviceRect(false, displayMatrix);
        PointF pt;
        float tolerance = 1.0;

        // Get the same annot (pAnnot) using annotRect.
        pt.x = annotRect.left + tolerance;
        pt.y = (annotRect.top - annotRect.bottom)/2 + annotRect.bottom;
        Annot gAnnot = page.GetAnnotAtDevicePoint(pt, tolerance, &displayMatrix);
        ...
}
```

3.14.1.5 *How to extract the texts under text markup annotations*

```
#include "include/common/fs_common.h"
```

```
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"
#include "include/pdf/fs_search.h"

using namespace std;
using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;
...

// Assuming PDFDoc doc has been loaded.
...

PDFPage page = doc.GetPage(0);
// Parse the first page.
page.StartParse(foxit::pdf::PDFPage::e_ParsePageNormal, NULL, false);
int annot_count = page.GetAnnotCount();
TextPage text_page(page);
for (int i = 0; i < annot_count; i++) {
	annots::Annot annot = page.GetAnnot(i);
	annots::TextMarkup text_markup(annot);
	if (!text_markup.IsEmpty()) {
		// Get the texts which intersect with a text markup annotation.
		WString text = text_page.GetTextUnderAnnot(text_markup);
	}
}
```

3.14.2 Import annotations from or export annotations to a FDF file

In Foxit PDF SDK, annotations can be created with data not only from applications but also from FDF files. At the same time, PDF SDK supports to export annotations to FDF files.

Example:

3.14.2.1 How to load annotations from a FDF file and add them into the first page of a given PDF

```
#include "include/common/fs_common.h"
#include "include/pdf/actions/fs_action.h"
#include "include/pdf/annots/fs_annot.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/objects/fs_pdfobject.h"
#include "include/pdf/fs_pdfpage.h"


using namespace std;
using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;
using namespace annots;

// Assuming PDFDoc doc has been loaded.
…

FILE* file = NULL;
#if defined(_WIN32) || defined(_WIN64)
      fopen_s(&file, (const char*)(const char*)String::FromUnicode(fdf_file), "rb+");
#else
      file = fopen((const char*)(const char*)String::FromUnicode(fdf_file), "rb+");
#endif
fseek(file, 0, SEEK_END);
size_t file_size = (size_t)ftell(file);
char* buffer = (char*)malloc(file_size * sizeof(char));
memset(buffer, 0 , file_size);

fseek(file, 0, SEEK_SET);
fread(buffer, sizeof(char), file_size, file);
fclose(file);

fdf::FDFDoc fdf_doc(buffer, file_size);
pdf_doc.ImportFromFDF(fdf_doc, PDFDoc::e_Annots);
```

3.15 Image Conversion

Foxit PDF SDK provides APIs for conversion between PDF files and images. Applications could easily fulfill functionalities like image creation and image conversion which supports the following image formats: BMP, TIFF, PNG, JPX, JPEG, and GIF. Foxit PDF SDK can make the conversion between PDF files and the supported image formats except for GIF. It only supports converting GIF images to PDF files.

Example:

3.15.1 How to convert PDF pages to bitmap files

```
#include "include/common/fs_common.h"
#include "include/common/fs_image.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"
#include "include/common/fs_render.h"

using namespace foxit;
using namespace foxit::common;
using namespace pdf;

// Assuming PDFDoc doc has been loaded.
…
```

```
// Get page count
int nPageCount = doc.GetPageCount();
for(int i=0;i<nPageCount;i++) {
    PDFPage page = doc.GetPage(i);

    // Parse page.
    page.StartParse(foxit::pdf::PDFPage::e_ParsePageNormal, NULL, false);

    int width = static_cast<int>(page.GetWidth());
    int height = static_cast<int>(page.GetHeight());
    Matrix matrix = page.GetDisplayMatrix(0, 0, width, height, page.GetRotation());

    // Prepare a bitmap for rendering.
    Bitmap bitmap(width, height, foxit::common::Bitmap::e_DIBArgb, NULL, 0);
    bitmap.FillRect(0xFFFFFFFF, NULL);

    // Render page.
    Renderer render(bitmap, false);
    render.StartRender(page, matrix, NULL);
        image.AddFrame(bitmap);
}
...
```

3.15.2 How to convert an image file to PDF file

```
#include "include/common/fs_common.h"
#include "include/common/fs_image.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"

using namespace foxit;
using namespace foxit::common;
using namespace pdf;


Image image(input_file);
int count = image.GetFrameCount();

PDFDoc doc;
for (int i = 0; i < count; i++) {
    PDFPage page = doc.InsertPage(i);
    page.StartParse(foxit::pdf::PDFPage::e_ParsePageNormal, NULL, false);
    // Add image to page.
    page.AddImage(image, i, PointF(0, 0), page.GetWidth(), page.GetHeight(), true);
}

doc.SaveAs(output_file, PDFDoc::e_SaveFlagNoOriginal);
...
```

3.16 Watermark

Watermark is a type of PDF annotation and is widely used in PDF document. Watermark is a visible embedded overlay on a document consisting of text, a logo, or a copyright notice. The purpose of a watermark is to identify the work and discourage its unauthorized use. Foxit PDF SDK provides APIs to work with watermark, allowing applications to create, insert, release and remove watermarks.

Example:

3.16.1 How to create a text watermark and insert it into the first page

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"
#include "include/pdf/fs_watermark.h"

using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;
...

// Assuming PDFDoc doc has been loaded.

WatermarkSettings settings;
settings.flags = WatermarkSettings::e_FlagASPageContents | WatermarkSettings::e_FlagOnTop;
settings.offset_x = 0;
settings.offset_y = 0;
settings.opacity = 90;
settings.position = common::e_PosTopRight;
settings.rotation = -45.f;
settings.scale_x = 1.f;
settings.scale_y = 1.f;

WatermarkTextProperties text_properties;
text_properties.alignment = CommonDefines::e_AlignmentCenter;
text_properties.color = 0xF68C21;
text_properties.font_style = WatermarkTextProperties::e_FontStyleNormal;
text_properties.line_space = 1;
text_properties.font_size = 12.f;
text_properties.font = Font(Font::e_StdIDTimesB);

Watermark watermark(doc, L"Foxit PDF SDK\nwww.foxitsoftware.com", text_properties, settings);
watermark.InsertToPage(doc.GetPage(0));

// Save document to file
...
```

3.16.2 How to create an image watermark and insert it into the first page

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"
#include "include/pdf/fs_watermark.h"

using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;
...

// Assuming PDFDoc doc has been loaded.

WatermarkSettings settings;
settings.flags = WatermarkSettings::e_FlagASPageContents | WatermarkSettings::e_FlagOnTop;
settings.offset_x = 0.f;
settings.offset_y = 0.f;
settings.opacity = 20;
settings.position = common::e_PosCenter;
settings.rotation = 0.0f;

Image image(image_file);
Bitmap bitmap = image.GetFrameBitmap(0);
settings.scale_x = page.GetWidth() * 0.618f / bitmap.GetWidth();
settings.scale_y = settings.scale_x;

Watermark watermark(doc, image, 0, settings);
watermark.InsertToPage(doc.GetPage(0));

// Save document to file.
...
```

3.16.3 How to remove all watermarks from a page

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"

using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;
...
// Assuming PDFPage page has been loaded and parsed.
...
page.RemoveAllWatermarks();
...
// Save document to file
```

```
...
```

3.17 Barcode

A barcode is an optical machine-readable representation of data relating to the object to which it is attached. Originally barcodes systematically represented data by varying the widths and spacing of parallel lines, and may be referred to as linear or one-dimensional (1D). Later they evolved into rectangles, dots, hexagons and other geometric patterns in two dimensions (2D). Although 2D systems use a variety of symbols, they are generally referred to as barcodes as well. Barcodes originally were scanned by special optical scanners called barcode readers. Later, scanners and interpretive software became available on devices including desktop printers and smartphones. Foxit PDF SDK provides applications to generate a barcode bitmap from a given string. The barcode types that Foxit PDF SDK supports are listed in Table 3-2.

Table 3-2

Barcode Type	**Code39**	**Code128**	**EAN8**	**UPCA**	**EAN13**	**ITF**	**PDF417**	**QR**
Dimension	1D	1D	1D	1D	1D	1D	2D	2D

Example:

3.17.1 How to generate a barcode bitmap from a string

```
#include "include/common/fs_common.h"
#include "include/common/fs_barcode.h"

using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
...

// Strings used as barcode content.
WString sz_code_string = L"TEST-SHEET";

// Barcode format types.
Barcode::Format code_format = Barcode::e_FormatCode39;

//Format error correction level of QR code.
Barcode::QRErrorCorrectionLevel sz_qr_level = Barcode::e_QRCorrectionLevelLow;

//Image names for the saved image files for QR code.
WString bmp_qr_name = L"/QR_CODE_TestForBarcodeQrCode_L.bmp";

// Unit width for barcode in pixels, preferred value is 1-5 pixels.
int unit_width = 2;
```

```
// Unit height for barcode in pixels, preferred value is >= 20 pixels.
int unit_height = 120;

Barcode barcode;
Bitmap bitmap = barcode.GenerateBitmap(sz_code_string, code_format, unit_width, unit_height, sz_qr_level);
...
```

3.18 Security

Foxit PDF SDK provides a range of encryption and decryption functions to meet different level of document security protection. Users can use regular password encryption and certificate-driven encryption, or using their own security handler for custom security implementation. It also provides APIs to integrate with the third-party security mechanism (Microsoft RMS). These APIs allow developers to work with the Microsoft RMS SDK to both encrypt (protect) and decrypt (unprotect) PDF documents.

***Note**: For more detailed information about the RMS encryption and decryption, please refer to the simple demo "**security**" in the* "\examples\simple_demo" folder of the download package.

Example:

3.18.1 How to encrypt a PDF file with Certificate

```
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_security.h"

using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;
...
PDFDoc doc(input_file);
ErrorCode error_code = doc.Load();
   if (error_code != foxit::e_ErrSuccess) {
     return false;
   }

// Do encryption.
StringArray envelopes;
String initial_key;
WString cert_file_path = input_path + L"foxit.cer";
if (!GetCertificateInfo((const char*)String::FromUnicode(cert_file_path), envelopes, initial_key, true, 16)) {
    return false;
}
CertificateSecurityHandler handler;
CertificateEncryptData encrypt_data(true, SecurityHandler::e_CipherAES, envelopes);
```

```
handler.Initialize(encrypt_data, initial_key);

doc.SetSecurityHandler(handler);
WString output_file = output_directory + L"certificate_encrypt.pdf";
doc.SaveAs(output_file, PDFDoc::e_SaveFlagNoOriginal);
...
```

3.18.2 How to encrypt a PDF file with Foxit DRM

```
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_security.h"

using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;
...

PDFDoc doc(input_file);
ErrorCode error_code = doc.Load();
if (error_code != foxit::e_ErrSuccess) {
  return false;
}

// Do encryption.
DRMSecurityHandler handler = DRMSecurityHandler();
const char* file_id = "Simple-DRM-file-ID";
String initialize_key = "Simple-DRM-initialize-key";
DRMEncryptData encrypt_data(true, "Simple-DRM-filter", SecurityHandler::e_CipherAES, 16, true, 0xfffffffc);
handler.Initialize(encrypt_data, file_id, initialize_key);
doc.SetSecurityHandler(handler);

WString output_file = output_directory + L"foxit_drm_encrypt.pdf";
doc.SaveAs(output_file, PDFDoc::e_SaveFlagNoOriginal);
...
```

3.19 Reflow

Reflow is a function that rearranges page content when the page size changes. It is useful for applications that have output devices with difference sizes. Reflow frees the applications from considering layout for different devices. This function provides APIs to create, render, release and access properties of 'reflow' pages.

Example:

3.19.1 How to create a reflow page and render it to a bmp file

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
```

```
#include "include/pdf/fs_pdfpage.h"
#include "include/pdf/fs_reflowpage.h"
#include "include/common/fs_render.h"

using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;
...

// Assuming PDFDoc doc has been loaded.

PDFPage page = doc.GetPage(0);
// Parse PDF page.
page.StartParse(foxit::pdf::PDFPage::e_ParsePageNormal, NULL, false);

ReflowPage reflow_page(page);

// Set some arguments used for parsing the reflow page.
reflow_page.SetLineSpace(0);
reflow_page.SetZoom(100);
reflow_page.SetParseFlags(ReflowPage::e_Normal);

// Parse reflow page.
reflow_page.StartParse(NULL);

// Get actual size of content of reflow page. The content size does not contain the margin.
float content_width = reflow_page.GetContentWidth();
float content_height = reflow_page.GetContentHeight();

// Assuming Bitmap bitmap has been created.

// Render reflow page.
Renderer renderer(bitmap, false);
foxit::Matrix matrix = reflow_page.GetDisplayMatrix(0, 0);
renderer.StartRenderReflowPage(reflow_page, matrix, NULL);
...
```

3.20 Asynchronous PDF

Asynchronous PDF technique is a way to access PDF pages without loading the whole document when it takes a long time. It's especially designed for accessing PDF files on internet. With asynchronous PDF technique, applications do not have to wait for the whole PDF file to be downloaded before accessing it. Applications can open any page when the data of that page is available. It provides a convenient and efficient way for web reading applications. For how to open and parse pages with asynchronous mode, you can refer to the simple demo "**async_load**" in the "\examples\simple_demo" folder of the download package.

3.21 Pressure Sensitive Ink

Pressure **S**ensitive **I**nk (**PSI**) is a technique to obtain varying electrical outputs in response to varying pressure or force applied across a layer of pressure sensitive devices. In PDF, PSI is usually used for hand writing signatures. PSI data are collected by touching screens or handwriting on boards. PSI data contains coordinates and canvas of the operating area which can be used to generate appearance of PSI. Foxit PDF SDK allows applications to create PSI, access properties, operate on ink and canvas, and release PSI.

Example:

3.21.1 How to create a PSI and set the related properties for it

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"
#include "include/pdf/fs_psi.h"

using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;
using namespace annots;

PSI psi(480, 180, true);

// Set ink diameter.
psi.SetDiameter(9);

// Set ink color.
psi.SetColor(0x434236);

// Set ink opacity.
psi.SetOpacity(0.8f);

// Add points to pressure sensitive ink.
float x = 121.3043f;
float y = 326.6846f;
float pressure = 0.0966f;
Path::PointType type = Path::e_TypeMoveTo;
psi.AddPoint(PointF(x, y), type, pressure);
...
```

3.22 Wrapper

Wrapper provides a way for users to save their own data related to a PDF document. For example, when opening an encrypted unauthorized PDF document, users may get an error message. In this case, users can still access wrapper data even when they do not have permissions to the PDF content.

The wrapper data could be used to provide information like where to get decryption method of this document.

Example:

3.22.1 How to open a document including wrapper data

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"
#include "include/common/fs_render.h"

using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;
using namespace common::file;

// file_name is PDF document which includes wrapper data.
PDFDoc doc(file_name);
ErrorCode code = doc.Load();
if (code != foxit::e_ErrSuccess) {
   return false;
}
if(!doc.IsWrapper()){
   return false;
}
int64 offset = doc.GetWrapperOffset();

FileReader file_reader(offset);
file_reader.LoadFile(String::FromUnicode(file_name));
…
```

3.23 PDF Objects

There are eight types of object in PDF: Boolean object, numerical object, string object, name object, array object, dictionary object, stream object and null object. PDF objects are document level objects that are different from page objects (see 3.24) which are associated with a specific page each. Foxit PDF SDK provides APIs to create, modify, retrieve and delete these objects in a document.

Example:

3.23.1 How to remove some properties from catalog dictionary

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/objects/fs_pdfobject.h"

using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;
using namespace objects;
...

// Assuming PDFDoc doc has been loaded.

PDFDictionary* catalog = doc.GetCatalog();
if (NULL == catalog) return;

const char* key_strings[] = { "Type", "Boolean", "Name", "String", "Array", "Dict"};
int count = sizeof(key_strings)/sizeof(key_strings[0]);
for (int i = 0; i < count; i ++) {
  if (catalog->HasKey(key_strings[i]))
      catalog->RemoveAt(key_strings[i]);
}
...
```

3.24 Page Object

Page object is a feature that allows novice users having limited knowledge of PDF objects (see 3.23 for details of PDF objects) to be able to work with text, path, image, and canvas objects. Foxit PDF SDK provides APIs to add and delete PDF objects in a page and set specific attributes. Using page object, users can create PDF page from object contents. Other possible usages of page object include adding headers and footers to PDF documents, adding an image logo to each page, or generating a template PDF on demand.

Example:

3.24.1 How to create a text object in a PDF page

```
#include "include/common/fs_common.h"
#include "include/pdf/graphics/fs_pdfgraphicsobject.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"


using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;
using namespace graphics;
...

// Assuming PDFPage page has been loaded and parsed.

POSITION position = page.GetLastGraphicsObjectPosition(GraphicsObject::e_TypeText);
TextObject* text_object = TextObject::Create();
```

```
text_object->SetFillColor(0xFFFF7F00);

// Prepare text state.
TextState state;
state.font_size = 80.0f;
state.font =  Font(L"Simsun", Font::e_StylesSmallCap, Font::e_CharsetGB2312, 0);
state.textmode = TextState::e_ModeFill;
text_object->SetTextState(page, state, false, 750);

// Set text.
text_object->SetText(L"Foxit Software");
POSITION last_position = page.InsertGraphicsObject(position, text_object);
...
```

3.24.2 How to add an image logo to a PDF page

```
#include "include/common/fs_common.h"
#include "include/pdf/graphics/fs_pdfgraphicsobject.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"

using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;
using namespace graphics;
...
//Assuming PDFPage page has been loaded and parsed.

POSITION position = page.GetLastGraphicsObjectPosition(GraphicsObject::e_TypeImage);
Image image(image_file);
ImageObject* image_object = ImageObject::Create(page.GetDocument());
image_object->SetImage(image, 0);

float width = static_cast<float>(image.GetWidth());
float height = static_cast<float>(image.GetHeight());

float page_width = page.GetWidth();
float page_height = page.GetHeight();

// Please notice the matrix value.
image_object->SetMatrix(Matrix(width, 0, 0, height, (page_width - width) / 2.0f, (page_height - height) / 2.0f));

page.InsertGraphicsObject(position, image_object);
page.GenerateContent();
...
```

3.25 Marked content

In PDF document, a portion of content can be marked as marked content element. Marked content helps to organize the logical structure information in a PDF document and enables stylized tagged PDF. Tagged PDF has a standard structure types and attributes that allow page content to be extracted and reused for other purposes. More details about marked content could be found in chapter 10.5 of PDF reference 1.7 [1].

Example:

3.25.1 How to get marked content in a page and get the tag name

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"
#include "include/common/fs_image.h"
#include "include/pdf/graphics/fs_pdfgraphicsobject.h"
#include "include/pdf/objects/fs_pdfobject.h"

using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;
using namespace graphics;
using namespace objects;

...
// Assuming PDFPage page has been loaded and parsed.

POSITION position = page.GetFirstGraphicsObjectPosition(GraphicsObject::e_TypeText);
TextObject* text_obj = reinterpret_cast<TextObject*>(page.GetGraphicsObject(position));
MarkedContent* content = text_obj->GetMarkedContent();
int item_count = content->GetItemCount();

// Get marked content property
for (int i = 0; i < item_count; i++) {
  String tag_name = content->GetItemTagName(i);
  int mcid = content->GetItemMCID(i);
}
...
```

3.26 Layer

PDF Layers, in other words, Optional Content Groups (OCG), are supported in Foxit PDF SDK. Users can selectively view or hide the contents in different layers of a multi-layer PDF document. Multi-layers are widely used in many application domains such as CAD drawings, maps, layered artwork and multi-language document, etc.

In Foxit PDF SDK, a PDF layer is associated with a layer node. To retrieve a layer node, user should construct a PDF LayerTree object first and then call function LayerTree::GetRootNode to get the root layer node of the whole layer tree. Furthermore, you can enumerate all the nodes in the layer tree from the root layer node. Foxit PDF SDK provides APIs to get/set layer data, view or hide the contents in different layers, set layers' name, add or remove layers, and edit layers.

Example:

3.26.1 How to create a PDF layer

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/common/fs_render.h"
#include "include/pdf/fs_pdflayer.h"

using namespace foxit;
using namespace pdf;
 ...
// Assuming PDFDoc doc has been loaded.

LayerTree layertree(doc);
LayerNode root = layertree.GetRootNode();
if (root.IsEmpty()) {
		printf("No layer information!\r\n");
		return ;
}
...
```

3.26.2 How to set all the layer nodes information

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/common/fs_render.h"
#include "include/pdf/fs_pdflayer.h"

using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;
 ...
// Assuming PDFDoc doc has been loaded.
```

```
void SetAllLayerNodesInformation(LayerNode layer_node) {
 if (layer_node.HasLayer()) {
   layer_node.SetDefaultVisible(true);
   layer_node.SetExportUsage(LayerTree::e_StateUndefined);
   layer_node.SetViewUsage(LayerTree::e_StateOFF);
   LayerPrintData print_data("subtype_print", LayerTree::e_StateON);
   layer_node.SetPrintUsage(print_data);
   LayerZoomData zoom_data(1, 10);
   layer_node.SetZoomUsage(zoom_data);
   WString new_name = WString(L"[View_OFF_Print_ON_Export_Undefined]") + layer_node.GetName();
   layer_node.SetName(new_name);
 }
 int count = layer_node.GetChildrenCount();
 for (int i = 0; i < count; i++) {
   LayerNode child = layer_node.GetChild(i);
   SetAllLayerNodesInformation(child);
 }
}

LayerTree layertree(doc);
LayerNode root = layertree.GetRootNode();
SetAllLayerNodesInformation(root);
...
```

3.26.3 How to edit layer tree

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/common/fs_render.h"
#include "include/pdf/fs_pdflayer.h"

using namespace foxit;
using namespace foxit::common;
using namespace pdf;
 ...
// Assuming PDFDoc doc has been loaded.

// edit layer tree
PDFDoc doc(input_file);
error_code = doc.Load();
layertree = LayerTree(doc);
root = layertree.GetRootNode();
int children_count = root.GetChildrenCount();
root.RemoveChild(children_count -1);
LayerNode child = root.GetChild(children_count - 2);
LayerNode child0 = root.GetChild(0);
child.MoveTo(child0, 0);
child.AddChild(0, L"AddedLayerNode", true);
child.AddChild(0, L"AddedNode", false);
```

3.27 Signature

PDF Signature module can be used to create and sign digital signatures for PDF documents, which protects the security of documents' contents and avoids it to be tampered maliciously. It can let the receiver make sure that the document is released by the signer and the contents of the document are complete and unchanged. Foxit PDF SDK provides APIs to create digital signature, verify the validity of signature, delete existing digital signature, get and set properties of digital signature, display signature and customize the appearance of the signature form fields.

***Note**: Foxit PDF SDK provides default Signature callbacks which supports the following two types of signature filter and subfilter:*

(1) filter: Adobe.PPKLite subfilter: adbe.pkcs7.detached

(2) filter: Adobe.PPKLite subfilter: adbe.pkcs7.sha1

If you use one of the above signature filter and subfilter, you can sign a PDF document and verify the validity of signature by default without needing to register a custom callback.

Example:

3.27.1 How to sigh the PDF document with a signature

```
#include "include/pdf/annots/fs_annot.h"
#include "include/common/fs_image.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"
#include "include/pdf/fs_signature.h"

using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;
using namespace objects;
using namespace file;

// AdobePPKLiteSignature
const char* filter = "Adobe.PPKLite";
const char* sub_filter = "adbe.pkcs7.detached";

if (!use_default) {
        InitializeOpenssl();
        sub_filter = "adbe.pkcs7.sha1";
        SignatureCallbackImpl* sig_callback = new SignatureCallbackImpl(sub_filter);
        Library::RegisterSignatureCallback(filter, sub_filter, sig_callback);
}

printf("Use signature callback object for filter \"%s\" and sub-filter \"%s\"\r\n",
            filter, sub_filter);
PDFPage pdf_page = pdf_doc.GetPage(0);
// Add a new signature to the first page.
Signature new_signature = AddSiganture(pdf_page, sub_filter);
// Set filter and subfilter for the new signature.
new_signature.SetFilter(filter);
new_signature.SetSubFilter(sub_filter);
bool is_signed = new_signature.IsSigned();
uint32 sig_state = new_signature.GetState();
printf("[Before signing] Signed?:%s\t State:%s\r\n",
            is_signed? "true" : "false",
            TransformSignatureStateToString(sig_state).c_str());

// Sign the new signature.
WString signed_pdf_path = output_directory + L"signed_newsignature.pdf";
if (use_default)
        signed_pdf_path = output_directory + L"signed_newsignature_default_handler.pdf";

WString cert_file_path = input_path + L"foxit_all.pfx";
WString cert_file_password = L"123456";
// Cert file path will be passed back to application through callback function FSSignatureCallback::Sign().
// In this demo, the cert file path will be used for signing in callback function FSSignatureCallback::Sign().
new_signature.StartSign((const wchar_t*)cert_file_path, cert_file_password,
            Signature::e_DigestSHA1, (const wchar_t*)signed_pdf_path, NULL, NULL);
printf("[Sign] Finished!\r\n");
is_signed = new_signature.IsSigned();
sig_state = new_signature.GetState();
printf("[After signing] Signed?:%s\tState:%s\r\n",
            is_signed? "true" : "false",
            TransformSignatureStateToString(sig_state).c_str());

// Open the signed document and verify the newly added signature (which is the last one).
printf("Signed PDF file: %s\r\n", (const char*)String::FromUnicode(signed_pdf_path));
PDFDoc signed_pdf_doc((const wchar_t*)signed_pdf_path);
ErrorCode error_code = signed_pdf_doc.Load(NULL);
if (foxit::e_ErrSuccess !=error_code ) {
        printf("Fail to open the signed PDF file.\r\n");
        return;
}
// Get the last signature which is just added and signed.
int sig_count = signed_pdf_doc.GetSignatureCount();
Signature signed_signature = signed_pdf_doc.GetSignature(sig_count-1);
// Verify the signature.
signed_signature.StartVerify(NULL, NULL);
printf("[Verify] Finished!\r\n");
is_signed = signed_signature.IsSigned();
sig_state = signed_signature.GetState();
printf("[After verifying] Signed?:%s\tState:%s\r\n",
            is_signed? "true" : "false",
            TransformSignatureStateToString(sig_state).c_str());
```

3.27.2 How to implement signature callback function of signing

```
#include "include/pdf/annots/fs_annot.h"
#include "include/common/fs_image.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"
#include "include/pdf/fs_signature.h"

using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;
using namespace objects;
using namespace file;

// Implementation of pdf::SignatureCallback
class SignatureCallbackImpl : public pdf::SignatureCallback {
public:
	SignatureCallbackImpl(string subfilter)
		: sub_filter_(subfilter)
		, digest_context_(NULL) {}
	~SignatureCallbackImpl();

	virtual void Release() {
		delete this;
	}
	virtual bool StartCalcDigest(const ReaderCallback* file, const uint32* byte_range_array,
		uint32 size_of_array, const Signature& signature, const void* client_data);
	virtual Progressive::State ContinueCalcDigest(const void* client_data, const PauseCallback* pause);
	virtual String GetDigest(const void* client_data);
	virtual String Sign(const void* digest, uint32 digest_length, const wchar_t* cert_path,
		const WString& password, Signature::DigestAlgorithm digest_algorithm,
		void* client_data);
	virtual uint32 VerifySigState(const void* digest, uint32 digest_length,
		const void* signed_data, uint32 signed_data_len,
		void* client_data);
	virtual bool IsNeedPadData() {return false;}
protected:
	bool GetTextFromFile(unsigned char *plainString);

	unsigned char* PKCS7Sign(const wchar_t* cert_file_path, String cert_file_password,
		String plain_text, int& signed_data_size);
	bool PKCS7VerifySignature(String signed_data, String plain_text);
	bool ParseP12File(const wchar_t* cert_file_path, String cert_file_password,
		EVP_PKEY** pkey, X509** x509, STACK_OF(X509)** ca);
	ASN1_INTEGER* CreateNonce(int bits);

private:
	string sub_filter_;
	DigestContext* digest_context_;

	string cert_file_path_;
	string cert_file_password_;
};

SignatureCallbackImpl::~SignatureCallbackImpl() {
	if (digest_context_) {
		delete digest_context_;
		digest_context_ = NULL;
	}
}

bool SignatureCallbackImpl::GetTextFromFile(unsigned char* file_buffer) {
	if (!digest_context_ || !digest_context_->GetFileReadCallback()) return false;
	ReaderCallback* file_read = digest_context_->GetFileReadCallback();
	file_read->ReadBlock(file_buffer, digest_context_->GetByteRangeElement(0),
digest_context_->GetByteRangeElement(1));
	file_read->ReadBlock(file_buffer + (digest_context_->GetByteRangeElement(1)-
digest_context_->GetByteRangeElement(0)),
		digest_context_->GetByteRangeElement(2), digest_context_->GetByteRangeElement(3));
	return true;
}

bool SignatureCallbackImpl::StartCalcDigest(const ReaderCallback* file, const uint32* byte_range_array,
	uint32 size_of_array, const Signature& signature, const void* client_data) {
		if (digest_context_) {
			delete digest_context_;
			digest_context_ = NULL;
		}
		digest_context_ = new DigestContext(const_cast<ReaderCallback*>(file), byte_range_array,
size_of_array);
		if(!SHA1_Init(&digest_context_->sha_ctx_)) {
			delete digest_context_;
			digest_context_ = NULL;
			return false;
		}
		return true;
}

Progressive::State SignatureCallbackImpl::ContinueCalcDigest(const void* client_data, const PauseCallback*
pause) {
	if (!digest_context_) return Progressive::e_Error;

	uint32 file_length = digest_context_->GetByteRangeElement(1) +
digest_context_->GetByteRangeElement(3);
	unsigned char* file_buffer = (unsigned char*)malloc(file_length);
	if (!file_buffer || !GetTextFromFile(file_buffer)) return Progressive::e_Error;

	SHA1_Update(&digest_context_->sha_ctx_, file_buffer, file_length);
	free(file_buffer);
	return Progressive::e_Finished;
}

String SignatureCallbackImpl::GetDigest(const void* client_data) {
        if (!digest_context_) return "";
        unsigned char* md = reinterpret_cast<unsigned
char*>(OPENSSL_malloc((SHA_DIGEST_LENGTH)*sizeof(unsigned char)));
        if (1 != SHA1_Final(md, &digest_context_->sha_ctx_))
                return "";
        String digest = String(reinterpret_cast<const char*>(md), SHA_DIGEST_LENGTH);
        OPENSSL_free(md);
        return digest;
}

String SignatureCallbackImpl::Sign(const void* digest, uint32 digest_length, const wchar_t* cert_path,
        const WString& password, Signature::DigestAlgorithm digest_algorithm,
        void* client_data) {
                if (!digest_context_) return "";
                String plain_text;
                if ("adbe.pkcs7.sha1" == sub_filter_) {
                        plain_text = String((const char*)digest, digest_length);
                }
                int signed_data_length = 0;
                unsigned char* signed_data_buffer = PKCS7Sign(cert_path, String::FromUnicode(password),
                        plain_text, signed_data_length);
                if (!signed_data_buffer) return "";

                String signed_data = String((const char*)signed_data_buffer, signed_data_length);
                free(signed_data_buffer);
                return signed_data;
}

uint32 SignatureCallbackImpl::VerifySigState(const void* digest, uint32 digest_length,
        const void* signed_data, uint32 signed_data_len, void* client_data) {
                // Usually, the content of a signature field is contain the certification of signer.
                // But we can't judge this certification is trusted.
                // For this example, the signer is ourself. So when using api PKCS7_verify to verify,
                // we pass NULL to it's parameter <i>certs</i>.
                // Meanwhile, if application should specify the certificates, we suggest pass flag
PKCS7_NOINTERN to
                // api PKCS7_verify.
                if (!digest_context_) return Signature::e_StateVerifyErrorData;
                String plain_text;
                unsigned char* file_buffer = NULL;
                if ("adbe.pkcs7.sha1" == sub_filter_) {
                        plain_text = String(reinterpret_cast<const char*>(digest), digest_length);
                } else {
                        return Signature::e_StateUnknown;
                }

                String signed_data_str = String(reinterpret_cast<const char*>(signed_data), signed_data_len);
                bool ret = PKCS7VerifySignature(signed_data_str, plain_text);
                if (file_buffer) free(file_buffer);
                return ret ? Signature::e_StateVerifyNoChange : Signature::e_StateVerifyChange;

}
```

```
ASN1_INTEGER* SignatureCallbackImpl::CreateNonce(int bits) {
        unsigned char buf[20];
        int len = (bits - 1) / 8 + 1;
        // Generating random byte sequence.
        if (len > (int)sizeof(buf)) {
                return NULL;
        }
        if (RAND_bytes(buf, len) <= 0) {
                return NULL;
        }
        // Find the first non-zero byte and creating ASN1_INTEGER object.
        int i = 0;
        for (i = 0; i < len && !buf[i]; ++i) ;
        ASN1_INTEGER* nonce = NULL;
        if (!(nonce = ASN1_INTEGER_new())) {
                ASN1_INTEGER_free(nonce);
                return NULL;
        }
        OPENSSL_free(nonce->data);
        // Allocate at least one byte.
        nonce->length = len - i;
        if (!(nonce->data = reinterpret_cast<unsigned char*>(OPENSSL_malloc(nonce->length + 1)))) {
                ASN1_INTEGER_free(nonce);
                return NULL;
        }
        memcpy(nonce->data, buf + i, nonce->length);
        return nonce;
}

bool SignatureCallbackImpl::ParseP12File(const wchar_t* cert_file_path, String cert_file_password,
        EVP_PKEY** pkey, X509** x509, STACK_OF(X509)** ca) {
                FILE* file = NULL;
#if defined(_WIN32) || defined(_WIN64)
                _wfopen_s(&file, cert_file_path, L"rb");
#else
                file = fopen(String::FromUnicode(cert_file_path), "rb");
#endif  // defined(_WIN32) || defined(_WIN64)
                if (!file) {
                        return false;
                }

                PKCS12* pkcs12 = d2i_PKCS12_fp(file, NULL);
                fclose (file);
                if (!pkcs12) {
                        return false;
                }

                if (!PKCS12_parse(pkcs12, (const char*)cert_file_password, pkey, x509, ca)) {
                        return false;
                }
                PKCS12_free(pkcs12);
                if (!pkey)
                        return false;
                return true;
}

unsigned char* SignatureCallbackImpl::PKCS7Sign(const wchar_t* cert_file_path, String cert_file_password,
        String plain_text, int& signed_data_size) {
                PKCS7* p7 = NULL;
                EVP_PKEY* pkey = NULL;
                X509* x509 = NULL;
                STACK_OF(X509)* ca = NULL;
                if(!ParseP12File(cert_file_path, cert_file_password, &pkey, &x509, &ca))
                        return NULL;

                p7 = PKCS7_new();
                PKCS7_set_type(p7, NID_pkcs7_signed);
                PKCS7_content_new(p7, NID_pkcs7_data);

                // Application should not judge the sign algorithm with the content's length.
                // Here, just for convenient;
                if (plain_text.GetLength() > 32)
                        PKCS7_ctrl(p7, PKCS7_OP_SET_DETACHED_SIGNATURE, 1, NULL);

                PKCS7_SIGNER_INFO* signer_info = PKCS7_add_signature(p7, x509, pkey, EVP_sha1());
                PKCS7_add_certificate(p7, x509);

                for (int i = 0; i< sk_num(CHECKED_STACK_OF(X509,ca)); i++)
                        PKCS7_add_certificate(p7, (X509*)sk_value(CHECKED_STACK_OF(X509,ca), i));

                // Set source data to BIO.
                BIO* p7bio = PKCS7_dataInit(p7, NULL);
                BIO_write(p7bio, plain_text.GetBuffer(1), plain_text.GetLength());
                PKCS7_dataFinal(p7, p7bio);

                FREE_CERT_KEY;
                BIO_free_all(p7bio);
                // Get signed data.
                unsigned long der_length = i2d_PKCS7(p7, NULL);
                unsigned char* der = reinterpret_cast<unsigned char*>(malloc(der_length));
                memset(der, 0, der_length);
                unsigned char* der_temp = der;
                i2d_PKCS7(p7, &der_temp);
                PKCS7_free(p7);
                signed_data_size = der_length;
                return (unsigned char*)der;
}

bool SignatureCallbackImpl::PKCS7VerifySignature(String signed_data, String plain_text) {
        // Retain PKCS7 object from signed data.
        BIO* vin = BIO_new_mem_buf((void*)signed_data.GetBuffer(1), signed_data.GetLength());
        PKCS7* p7 = d2i_PKCS7_bio(vin, NULL);
```

```
	STACK_OF(PKCS7_SIGNER_INFO) *sk = PKCS7_get_signer_info(p7);
	int sign_count = sk_PKCS7_SIGNER_INFO_num(sk);

	int length = 0;
	bool bSigAppr = false;
	unsigned char *p = NULL;
	for(int i=0;i<sign_count; i++) {
		PKCS7_SIGNER_INFO* sign_info = sk_PKCS7_SIGNER_INFO_value(sk,i);

		BIO *p7bio = BIO_new_mem_buf((void*)plain_text.GetBuffer(1), plain_text.GetLength());
		X509 *x509= PKCS7_cert_from_signer_info(p7,sign_info);
		if(1 == PKCS7_verify(p7, NULL, NULL,p7bio, NULL, PKCS7_NOVERIFY))
			bSigAppr = true;
		BIO_free(p7bio);
	}
	PKCS7_free(p7);
	BIO_free(vin);
	return bSigAppr;
}
```

3.28 Long term validation (LTV)

From version 7.0, Foxit PDF SDK provides APIs to establish long term validation of signatures, which is mainly used to solve the verification problem of signatures that have already expired. LTV requires DSS (Document Security Store) which contains the verification information of the signatures, as well as DTS (Document Timestamp Signature) which belongs to the type of time stamp signature.

In order to support LTV, Foxit PDF SDK provides:

- Support for adding the signatures of time stamp type, and provides a default signature callback for the subfilter "ETSI.RFC3161".
- TimeStampServerMgr and TimeStampServer classes, which are used to set and manager the server for time stamp. The default signature callback for the subfilter "ETSI.RFC3161" will use the default time stamp server.
- LTVVerifier class which offers the functionalities of verifying signatures and adding DSS information to documents. It also provides a basic default RevocationCallback which is required by LTVVerifier.

Following lists an example about how to establish long term validation of signatures using the default signature callback for subfilter "ETSI.RFC3161" and the default RevocationCallback. For more details, please refer to the simple demo "**ltv**" in the "\examples\simple_demo" folder of the download package.

Example:

3.28.1 How to establish long term validation of signatures using the default signature callback for subfilter "ETSI.RFC3161" and the default RevocationCallback

```
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"
#include "include/pdf/fs_signature.h"
#include "include/pdf/fs_ltvverifier.h"
...

// Initialize time stamp server manager, add and set a default time stamp server, which will be used by default
signature callback for time stamp signature.
TimeStampServerMgr::Initialize();
TimeStampServer timestamp_server = TimeStampServerMgr::AddServer(server_name, server_url,
server_username, server_password);
TimeStampServerMgr::SetDefaultServer(timestamp_server);

// Assume that "signed_pdf_path" represents a signed PDF document which contains signed signature.
PDFDoc pdf_doc(signed_pdf_path);
pdf_doc.StartLoad();
{
        // Use LTVVerifier to verify and add DSS.
        LTVVerifier ltv_verifier(pdf_doc, true, false, false, LTVVerifier::e_SignatureTSTTime);
        // Set verifying mode which is necessary.
        ltv_verifier.SetVerifyMode(LTVVerifier::e_VerifyModeETSI);
        SignatureVerifyResultArray sig_verify_result_array = ltv_verifier.Verify();
        for (size_t i = 0; i < sig_verify_result_array.GetSize(); i++) {
                // ltv state would be e_LTVStateNotEnable here.
                SignatureVerifyResult::LTVState ltv_state =  sig_verify_result_array.GetAt(i).GetLTVState();
                if (sig_verify_result_array.GetAt(i).GetSignatureState() & Signature::e_StateVerifyValid)
                        ltv_verifier.AddDSS(sig_verify_result_array.GetAt(i));
        }
}

// Add a time stamp signature as DTS and sign it. "saved_ltv_pdf_path" represents the newly saved signed PDF
file.
PDFPage pdf_page = pdf_doc.GetPage(0);
// The new time stamp signature will have default filter name "Adobe.PPKLite" and default subfilter name
"ETSI.RFC3161".
Signature timestamp_signature = pdf_page.AddSignature(RectF(), L"", Signature::e_SignatureTypeTimeStamp);
Progressive sign_progressive = timestamp_signature.StartSign(L"", L"", Signature::e_DigestSHA256,
saved_ltv_pdf_path);
if (sign_progressive.GetRateOfProgress() != 100)
        sign_progressive.Continue();

// Then use LTVVeirfier to verify the new signed PDF file.
PDFDoc check_pdf_doc(saved_ltv_pdf_path);
check_pdf_doc.StartLoad();
{
        // Use LTVVeirfier to verify.
        LTVVerifier ltv_verifier(pdf_doc, true, false, false, LTVVerifier::e_SignatureTSTTime);
        // Set verifying mode which is necessary.
        ltv_verifier.SetVerifyMode(LTVVerifier::e_VerifyModeETSI);
        SignatureVerifyResultArray sig_verify_result_array = ltv_verifier.Verify();
        for (size_t i = 0; i < sig_verify_result_array.GetSize(); i++) {
                // ltv state would be e_LTVStateEnable here.
                SignatureVerifyResult::LTVState ltv_state =  sig_verify_result_array.GetAt(i).GetLTVState();
                ... // User can get other information from SignatureVerifyResult.
        }
}

// Release time stamp server manager when everything is done.
TimeStampServerMgr::Release();
```

3.29 PAdES

From version 7.0, Foxit PDF SDK also supports PAdES (PDF Advanced Electronic Signature) which is the application for CAdES signature in the field of PDF. CAdES is a new standard for advanced digital signature, its default subfilter is "ETSI.CAdES.detached". PAdES signature includes four levels: B-B, B-T, B-LT, and B-LTA.

- B-B: Must include the basic attributes.
- B-T: Must include document time stamp or signature time stamp to provide trusted time for existing signatures, based on B-B.
- B-LT: Must include DSS/VRI to provide certificates and revocation information, based on B-T.
- B-LTA: Must include the trusted time DTS for existing revocation information, based on B-LT.

Foxit PDF SDK provides a default signature callback for the subfilter "ETSI.CAdES.detached" to sign and verify the signatures (with subfilter "ETSI.CAdES.detached"). It also provides TimeStampServerMgr and TimeStampServer classes to set and manager the server for time stamp. The default signature callback for the subfilter "ETSI.CAdES.detached" will use the default time stamp server.

Foxit PDF SDK provides functions to get the level of PAdES from signature, and application level can also judge and determine the level of PAdES according to the requirements of each level. For more details about how to add, sign and verify a PAdES signature in PDF document, please refer to the simple demo "**pades**" in the "\examples\simple_demo" folder of the download package.

3.30 PDF Action

PDF Action is represented as the base PDF action class. Foxit PDF SDK provides APIs to create a series of actions and get the action handlers, such as embedded goto action, JavaScript action, named action and launch action, etc.

Example:

3.30.1 How to create a URI action and insert to a link annot

```
#include "include/common/fs_common.h"
#include "include/pdf/actions/fs_action.h"
#include "include/pdf/annots/fs_annot.h"
#include "include/pdf/objects/fs_pdfobject.h"

using namespace foxit;
using namespace foxit::common;
using namespace pdf;
using namespace annots;

// Assuming PDFPage page has been loaded and parsed.
// Assuming the annnots in the page have been loaded.
...

// Add link annotation
annots::Link link(page.AddAnnot(Annot::e_Link, RectF(350,350,380,400)));
link.SetHighlightingMode(Annot::e_HighlightingToggle);
// Add action for link annotation
using foxit::pdf::actions::Action;
using foxit::pdf::actions::URIAction;
URIAction action = (URIAction)Action::Create(page.GetDocument(), Action::e_TypeURI);
action.SetTrackPositionFlag(true);
action.SetURI("www.foxitsoftware.com");
link.SetAction(action);
// Appearance should be reset.
link.ResetAppearanceStream();
```

3.30.2 How to create a GoTo action and insert to a link annot

```
#include "include/common/fs_common.h"
#include "include/pdf/actions/fs_action.h"
#include "include/pdf/annots/fs_annot.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/objects/fs_pdfobject.h"
#include "include/pdf/fs_pdfpage.h"

using namespace foxit;
using namespace foxit::common;
using namespace pdf;
using namespace annots;
```

```
// Assuming the PDFDoc doc has been loaded.
// Assuming PDFPage page has been loaded and parsed.

// Add link annotation
annots::Link link(page.AddAnnot(Annot::e_Link, RectF(350,350,380,400)));
link.SetHighlightingMode(Annot::e_HighlightingToggle);

GotoAction action = Action::Create(page.GetDocument(), Action::e_TypeGoto);
Destination newDest = Destination::CreateXYZ(page.GetDocument(), 0,0,0,0);
action.SetDestination(newDest);
```

3.31 JavaScript

JavaScript was created to offload Web page processing from a server onto a client in Web-based applications. Foxit PDF SDK JavaScript implements extensions, in the form of new objects and their accompanying methods and properties, to the JavaScript language. It enables a developer to manage document security, communicate with a database, handle file attachments, and manipulate a PDF file so that it behaves as an interactive, web-enabled form, and so on.

JavaScript action is an action that causes a script to be compiled and executed by the JavaScript interpreter. Class foxit::pdf::actions::JavaScriptAction is derived from Action and offers functions to get/set JavaScript action data.

The JavaScript methods and properties supported by Foxit PDF SDK are listed in the appendix.

Example:

3.31.1 How to add JavaScript Action to Document

```
#include "include/pdf/actions/fs_action.h"
#include "include/pdf/annots/fs_annot.h"

using namespace foxit;
using namespace pdf;
using namespace annots;

// Load Document doc.
...

actions::JavaScriptAction javascript_action = (actions::JavaScriptAction)Action::Create(doc,
Action::e_TypeJavaScript);
javascipt_action.SetScript(L"app.alert(\"Hello Foxit \");");
AdditionalAction additional_act(doc);
```

```
additional_act.SetAction(AdditionalAction::e_TriggerDocWillClose,javascipt_action);
additional_act.DoJSAction(AdditionalAction::e_TriggerDocWillClose);
...
```

3.31.2 How to add JavaScript Action to Annotation

```
#include "include/pdf/actions/fs_action.h"
#include "include/pdf/annots/fs_annot.h"

using namespace foxit;
using namespace pdf;
using namespace annots;

// Load Document and get a widget annotation.
...

actions::JavaScriptAction javascript_action = (actions::JavaScriptAction)Action::Create(page.GetDocument(),
Action::e_TypeJavaScript);
javascipt_action.SetScript(L"app.alert(\"Hello Foxit \");");
AdditionalAction additional_act(annot);
additional_act.SetAction(AdditionalAction::e_TriggerAnnotMouseButtonPressed,javascipt_action);
additional_act.DoJSAction(AdditionalAction::e_TriggerAnnotMouseButtonPressed);
...
```

3.31.3 How to add JavaScript Action to FormField

```
#include "include/pdf/actions/fs_action.h"
#include "include/pdf/annots/fs_annot.h"
#include "include/pdf/interform/fs_pdfform.h"

using namespace foxit;
using namespace pdf;
using namespace annots;
using namespace interform;

// Load Document and get a form field.
...

// Add text field.
Control control = form.AddControl(page, L"Text Field0", Field::e_TypeTextField, RectF(50, 600, 90, 640));
control.GetField().SetValue(L"3");
// Update text field's appearance.
control.GetWidget().ResetAppearanceStream();
```

```
Control control1 = form.AddControl(page, L"Text Field1", Field::e_TypeTextField, RectF(100, 600, 140, 640));
control1.GetField().SetValue(L"23");
// Update text field's appearance.
control1.GetWidget().ResetAppearanceStream();

Control control2 = form.AddControl(page, L"Text Field2", Field::e_TypeTextField, RectF(150, 600, 190, 640));
actions::JavaScriptAction javascipt_action = (actions::JavaScriptAction)Action::Create(form.GetDocument(),
Action::e_TypeJavaScript);
javascipt_action.SetScript(L"AFSimple_Calculate(\"SUM\", new Array (\"Text Field0\", \"Text Field1\"));" );
Field field2 = control2.GetField();
AdditionalAction additional_act(field2);
additional_act.SetAction(AdditionalAction::e_TriggerFieldRecalculateValue,javascipt_action);
// Update text field's appearance.
control2.GetWidget().ResetAppearanceStream();
...
```

3.31.4 How to add a new annotation to PDF using JavaScript

```
#include "include/pdf/actions/fs_action.h"
#include "include/pdf/annots/fs_annot.h"


// Load Document and get form field, construct a Form object and a Filler object.
...

actions::JavaScriptAction javascipt_action = (actions::JavaScriptAction)Action::Create(form.GetDocument(),
Action::e_TypeJavaScript);
javascipt_action.SetScript(L"var annot = this.addAnnot({ page : 0, type : \"Square\", rect : [ 0, 0, 100, 100 ], name :
\"UniqueID\", author : \"A. C. Robat\", contents : \"This section needs revision.\" });" );
AdditionalAction additional_act(field);
additional_act.SetAction(AdditionalAction::e_TriggerAnnotCursorEnter,javascipt_action);
additional_act.DoJSAction(AdditionalAction::e_TriggerAnnotCursorEnter);
...
```

3.31.5 How to get/set properties of annotations (strokeColor, fillColor, readOnly, rect, type) using JavaScript

```
#include "include/pdf/actions/fs_action.h"
#include "include/pdf/annots/fs_annot.h"


// Load Document and get form field, construct a Form object and a Filler object.
...

// Get properties of annotations.
actions::JavaScriptAction javascipt_action = (actions::JavaScriptAction)Action::Create(form.GetDocument(),
Action::e_TypeJavaScript);
```

```
javascipt_action.SetScript(L"var ann = this.getAnnot(0, \" UniqueID \"); if (ann != null) { console.println(\"Found it! type: \" + ann.type); console.println(\"readOnly: \" + ann.readOnly); console.println(\"strokeColor: \" + ann.strokeColor);console.println(\"fillColor: \" + ann.fillColor); console.println(\"rect: \" + ann.rect);}");
AdditionalAction additional_act(field);
additional_act.SetAction(AdditionalAction::e_TriggerAnnotCursorEnter,javascipt_action);
additional_act.DoJSAction(AdditionalAction::e_TriggerAnnotCursorEnter);

// Set properties of annotations (only take strokeColor as an example).
actions::JavaScriptAction javascipt_action1 = (actions::JavaScriptAction)Action::Create(form.GetDocument(), Action::e_TypeJavaScript);
javascipt_action1.SetScript(L"var ann = this.getAnnot(0, \"UniqueID\");if (ann != null) { ann.strokeColor = color.blue; }");
AdditionalAction additional_act1(field1);
additional_act1.SetAction(AdditionalAction::e_TriggerAnnotCursorEnter,javascipt_action1);
additional_act1.DoJSAction(AdditionalAction::e_TriggerAnnotCursorEnter);
...
```

3.31.6 How to destroy annotation using JavaScript

```
#include "include/pdf/actions/fs_action.h"
#include "include/pdf/annots/fs_annot.h"

// Load Document and get form field, construct a Form object and a Filler object.
...

actions::JavaScriptAction javascipt_action = (actions::JavaScriptAction)Action::Create(form.GetDocument(), Action::e_TypeJavaScript);
javascipt_action.SetScript(L"var ann = this.getAnnot(0, \" UniqueID \"); if (ann != null) { ann.destroy(); } " );
AdditionalAction additional_act(field);
additional_act.SetAction(AdditionalAction::e_TriggerAnnotCursorEnter,javascipt_action);
additional_act.DoJSAction(AdditionalAction::e_TriggerAnnotCursorEnter);
...
```

3.32 Redaction

Redaction is the process of removing sensitive information while keeping the document's layout. It allows users to permanently remove (redact) visible text and images from PDF documents to protect confidential information, such as social security numbers, credit card information, product release dates, and so on.

Redaction is a type of markup annotation, which is used to mark some contents of a PDF file and then the contents will be removed once the redact annotations are applied.

To do Redaction, you can use the following APIs:

- Call function foxit::addon::Redaction::Redaction to create a redaction module. If module "Redaction" is not defined in the license information which is used in function common::Library::Initialize, it means user has no right in using redaction related functions and this constructor will throw exception foxit::e_ErrInvalidLicense.
- Then call function foxit::addon::Redaction::MarkRedactAnnot to create a redaction object and mark page contents (text object, image object, and path object) which are to be redacted.
- Finally call function foxit::addon::Redaction::Apply to apply redaction in marked areas: remove the text or graphics under marked areas permanently.

Note*: To use the redaction feature, please make sure the license key has the permission of the 'Redaction' module.*

Example:

3.32.1 How to redact the text "PDF" on the first page of a PDF

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"
#include "include/pdf/fs_search.h"
#include "include/addon/fs_redaction.h"
#include "include/common/fs_render.h"

using namespace foxit;
using namespace common;
using namespace addon;
using namespace pdf;
using namespace foxit::pdf::annots;
...

Redaction redaction(doc);
// Parse PDF page.
PDFPage page = doc.GetPage(0);
page.StartParse(foxit::pdf::PDFPage::e_ParsePageNormal, NULL, false);
//Find Text Object to redact
TextPage text_page(page);
TextSearch text_search(text_page);
text_search.SetPattern(L"PDF");
RectFArray rect_array;
while(text_search.FindNext()) {
		rect_array.Append(text_search.GetMatchRects());
 }
if(rect_array.GetSize() > 0) {
		Redact redact = redaction.MarkRedactAnnot(page, rect_array);
		redact.ResetAppearanceStream();
```

```
            doc.SaveAs(output_directory + L"AboutFoxit_redected_default.pdf");

            // set border color to Green
            redact.SetBorderColor((long)0x00FF00);
            // set fill color to Blue
            redact.SetFillColor((long)0x0000FF);
            // set rollover fill color to Red
            redact.SetApplyFillColor((long)0xFF0000);
            redact.ResetAppearanceStream();
            doc.SaveAs(output_directory + L"AboutFoxit_redected_setColor.pdf");

            redact.SetOpacity((float)0.5);
            redact.ResetAppearanceStream();
            doc.SaveAs(output_directory + L"AboutFoxit_redected_setOpacity.pdf");

            if(redaction.Apply())
                        cout << "Redact page(0) succeed." << endl;
            else
                        cout << "Redact page(0) failed." << endl;
 }
doc.SaveAs(output_directory + L"AboutFoxit_redected_apply.pdf");
```

3.33 Comparison

Comparison feature lets you see the differences in two versions of a PDF. Foxit PDF SDK provides APIs to compare two PDF documents page by page, the differences between the two documents will be returned. For now, it only supports comparing the text of the PDF document, and in the next release, more PDF elements will be supported.

The differences can be defined into three types: delete, insert and replace. You can save these differences into a PDF file and mark them as annotations.

***Note**: To use the comparison feature, please make sure the license key has the permission of the 'Comparison' module*

Example:

3.33.1 How to compare two PDF documents and save the differences between them into a PDF file

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"
#include "include/pdf/fs_search.h"
#include "include/addon/fs_compare.h"
#include "include/common/fxcrt/fx_basic.h"

using namespace foxit;
using namespace common;
using namespace addon;
using namespace pdf;
using namespace foxit::pdf::annots;
...

PDFDoc base_doc("input_base_file");
ErrorCode error_code = base_doc.Load();
if (error_code != foxit::e_ErrSuccess) {
   return 1;
}

PDFDoc compared_doc("input_compared_file");
error_code = compared_doc.Load();
if (error_code != foxit::e_ErrSuccess) {
   return 1;
}

Comparison comparison(base_doc, compared_doc);

// Start comparing.
CompareResults result = comparison.DoCompare(0, 0, Comparison::e_CompareTypeText);
CompareResultInfoArray& oldInfo = result.results_base_doc;
CompareResultInfoArray& newInfo = result.results_compared_doc;
int oldInfoSize = oldInfo.GetSize();
int newInfoSize = newInfo.GetSize();
PDFPage page = compared_doc.GetPage(0);
for (int i=0; i<newInfoSize; i++)
{
   const CompareResultInfo& item = newInfo.GetAt(i);
   CompareResultInfo::CompareResultType type = item.type;
   if (type == CompareResultInfo::e_CompareResultTypeDeleteText)
   {
      WString res_string;
      res_string.Format((FX_LPCWSTR)L"\"%S\"", (FX_LPCWSTR)item.diff_contents);

      // Add stamp to mark the "delete" type differences between the two documents.
      CreateDeleteTextStamp(page, item.rect_array, 0xff0000, res_string, L"Compare : Delete", L"Text");
   }
   else if (type == CompareResultInfo::e_CompareResultTypeInsertText)
   {
      WString res_string;
      res_string.Format((FX_LPCWSTR)L"\"%S\"", (FX_LPCWSTR)item.diff_contents);

      // Highlight the "insert" type differences between the two documents.
      CreateHighlightRect(page, item.rect_array, 0x0000ff, res_string, L"Compare : Insert", L"Text");
   }
   else if (type == CompareResultInfo::e_CompareResultTypeReplaceText)
   {
      WString res_string;
      res_string.Format((FX_LPCWSTR)L"[Old]: \"%S\"\r\n[New]: \"%S\"",
(FX_LPCWSTR)oldInfo.GetAt(i).diff_contents, (FX_LPCWSTR)item.diff_contents);
```

```
        // Highlight the "replace" type differences between the two documents.
        CreateHighlightRect(page, item.rect_array, 0xe7651a, res_string, L"Compare : Replace", L"Text");
    }
}

// Save the comparison result to a PDF file.
compared_doc.SaveAs(output_directory + L"result.pdf");
```

***Note**: for CreateDeleteTextStamp and CreateHighlightRect functions, please refer to the simple demo "**pdfcompare**" located in the "\examples\simple_demo" folder of the download package.*

3.34 OCR

Optical Character Recognition, or OCR, is a software process that enables images or printed text to be translated into machine-readable text. OCR is most commonly used when scanning paper documents to create electronic copies, but can also be performed on existing electronic documents (e.g. PDF).

This section will provide instructions on how to set up your environment for the OCR feature module using Foxit PDF SDK for Windows.

3.34.1 System requirements

Platform: Windows

Programming Language: C++, Java, C#

License Key requirement: 'OCR' module permission in the license key

SDK Version: Foxit PDF SDK for Windows (C++, Java, C#) 6.4 or higher

3.34.2 Trial limit for SDK OCR add-on module

For the trial version, there are three trail limits that you should notice:

1) Allow 30 consecutive natural days to evaluate SDK from the first time of OCREngine initialization.
2) Allow up to 5000 pages to be converted using OCR from the first time of OCREngine initialization.
3) Trail watermarks will be generated on the PDF pages. This limit is used for all of the SDK modules.

3.34.3 OCR resource files

Please contact Foxit support team or sales team to get the OCR resource files package.

After getting the package, extract it to a desired directory (for example, extract the package to a directory named "ocr_addon"), and then you can see the resource files for OCR are as follows:

- **debugging_files:** Resource files used for debugging the OCR project. These file(s) cannot be distributed.
- **language_resource_CJK:** Resource files for CJK language, including: Chinese-Simplified, Chinese-Traditional, Japanese, and Korean.
- **language_resources_noCJK:** Resource files for the languages except CJK, including: Basque, Bulgarian, Catalan, Croatian, Czech, Danish, Dutch, English , Estonian, Faeroese, Finnish, French, Galician, German, Greek, Hebrew, Hungarian, Icelandic, Italian, Latvian(Lettish), Lithuanian, Macedonian, Maltese, Norwegian, Polish, Portuguese, Romanian, Russian, Serbian, Slovak, Slovenian, Spanish, Swedish, Thai, Turkish, Ukrainian.
- **win32_lib:** 32-bit library resource files
- **win64_lib:** 64-bit library resource files
- **readme.txt:** A txt file for introducing the role of each folder in this directory, as well as how to use those resource files for OCR.

3.34.4 How to run the OCR demo

Foxit PDF SDK for Windows provides an OCR demo located in the "\examples\simple_demo\ocr" folder to show you how to use Foxit PDF SDK to do OCR for a PDF page or a PDF document.

3.34.4.1 Load the OCR demo in Visual Studio

To load the OCR demo in your specific environment, please choose one of the following ways:

1) Load the visual studio solution files "simple_demo_vs2010.sln" or "simple_demo_vs2015.sln" or "simple_demo_vs2017.sln" (depending on your Visual Studio version) in the "\examples\simple_demo" folder. Right-click the ocr demo, choose **Set as StartUp Project**.
2) Load the "ocr_vs2010.vcxproj" or "ocr_vs2015.vcxproj" or "ocr_vs2017.vcxproj" (depending on your Visual Studio version) in the "\examples\simple_demo\ocr" folder.

3.34.4.2 Build an OCR resource directory

Before running the OCR demo, you should first build an OCR resource directory, and then pass the directory to Foxit PDF SDK API **OCREngine::Initialize** to initialize OCR engine.

To build an OCR resource directory, please follow the steps below:

1) Create a new folder to add the resources. For example, "D:/ocr_resources".

2) Add the appropriate library resource based on the platform architecture.

   - For **win32**, copy **all the files** under "ocr_addon/win32_lib" folder to "D:/ocr_resources".
   - For **win64**, copy **all the files** under "ocr_addon/win64_lib" folder to "D:/ocr_resources".

3) Add the language resource.

   - For CJK (Chinese-Simplified, Chinese-Traditional, Japanese, and Korean), copy **all the files** under "ocr_addon/language_resource_CJK" folder to "D:/ocr_resources".
   - For all other languages except CJK, copy **all the files** under "ocr_addon/language_resources_noCJK" folder to "D:/ocr_resources".
   - For all the supported languages, copy **all the files** under "ocr_addon/language_resource_CJK" and "ocr_addon/language_resources_noCJK" folders to "D:/ocr_resources".

4) (Optional) Add debugging file resource if you need to debug the demo.

   - For win32, copy the file(s) under "ocr_addon/debugging_files/win32" folder to "D:/ocr_resources".
   - For win64, copy the file(s) under "ocr_addon/debugging_files/win64" folder to "D:/ocr_resources".

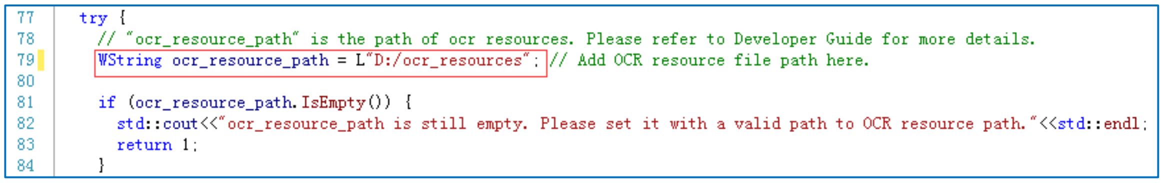
   ***Note**: The debugging files should be exclusively used for testing purposes. So, you cannot distribute them.*

3.34.4.3 Configure the demo

After building the OCR resource directory, configure the demo in the "\examples\simple_demo\ocr\ocr.cpp" file.

Specify the OCR resource directory

Add the OCR resource directory as follows, which will be used to initialize the OCR engine.



```
try {
  // "ocr_resource_path" is the path of ocr resources. Please refer to Developer Guide for more details.
  WString ocr_resource_path = L"D:/ocr_resources"; // Add OCR resource file path here.

  if (ocr_resource_path.IsEmpty()) {
    std::cout<<"ocr_resource_path is still empty. Please set it with a valid path to OCR resource path."<<std::endl;
    return 1;
  }
```

Choose the language resource

You will need to set the language used by the OCR engine into the demo code. This is done with the **OCREngine::SetLanguages** method and is set to "English" by default.

```
    // Set languages.
    OCREngine::SetLanguages(L"English");
```

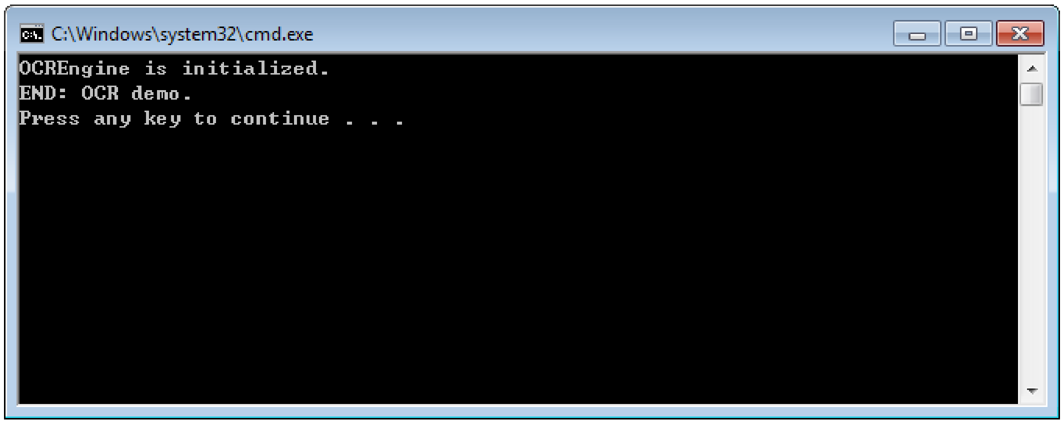
(Optional) Set log for OCREngine

If you add the debugging file resource to the OCR resource directory, and want to print the entire log of the OCR Engine, please uncomment the **OCREngine::SetLogFile** method as below:

```
    // Set log for OCREngine. (This can be opened to set log file if necessary)
    OCREngine::SetLogFile(output_directory+L"ocr.log");
```

3.34.4.4 Run the demo

Once you run the demo successfully, the console will print the following by default:



```
OCREngine is initialized.
END: OCR demo.
Press any key to continue . . .
```

The demo will OCR the default document ("\examples\simple_demo\input_files\ocr\AboutFoxit_ocr.pdf") in four different ways, which will output four different PDFs in the output folder ("\examples\simple_demo\output_files\ocr"):

- OCR Editable PDF - ocr_doc_editable.pdf
- OCR Searchable PDF - ocr_doc_searchable.pdf
- OCR Editable PDF Page - ocr_page_editable.pdf
- OCR Searchable PDF Page - ocr_page_searchable.pdf

3.35 Compliance

PDF Compliance

Foxit PDF SDK supports to convert PDF versions among PDF 1.3, PDF 1.4, PDF 1.5, PDF 1.6 and PDF 1.7. When converting to PDF 1.3, if the source document contains transparency data, then it will be converted to PDF 1.4 instead of PDF 1.3 (PDF 1.3 does not support transparency). If the source document does not contain any transparency data, then it will be converted to PDF 1.3 as expected.

PDF/A Compliance

PDF/A is an ISO-standardized version of the PDF specialized for use in the archiving and long-term preservation of electronic documents. PDF/A differs from PDF by prohibiting features unsuitable for long-term archiving, such as font linking (as opposed to font embedding), encryption, JavaScript, audio, video and so on.

Foxit PDF SDK provides APIs to convert a PDF to be compliance with PDF/A standard, or verify whether a PDF is compliance with PDF/A standard. It supports the PDF/A version including PDF/A-1a, PDF/A-1b, PDF/A-2a, PDF/A-2b, PDF/A-2u, PDF/A-3a, PDF/A-3b, PDF/A-3u (ISO 19005- 1, 19005 -2 and 19005-3).

This section will provide instructions on how to set up your environment for running the 'compliance' demo.

3.35.1 System requirements

Platform: Windows, Linux, Mac

Programming Language: C++, Java, C#, Objective-C

License Key requirement: 'Compliance' module permission in the license key

SDK Version: Foxit PDF SDK 6.4 or higher (for PDF Compliance, it requires Foxit PDF SDK 7.1 or higher)

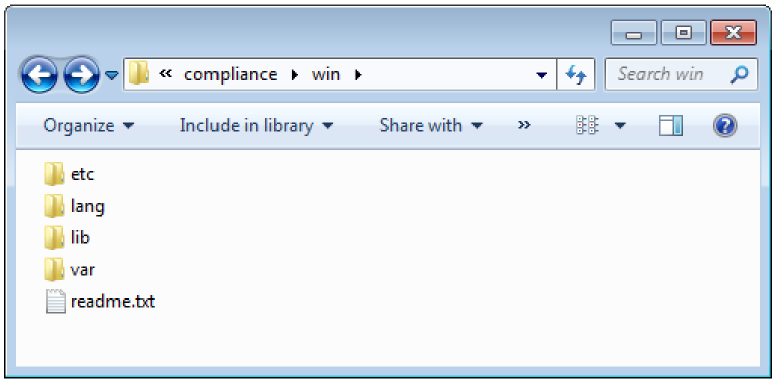
3.35.2 Compliance resource files

Please contact Foxit support team or sales team to get the Compliance resource files package.

After getting the package, extract it to a desired directory (for example, extract the package to a directory named "compliance"), and then you can see the resource files for Compliance are as follows:

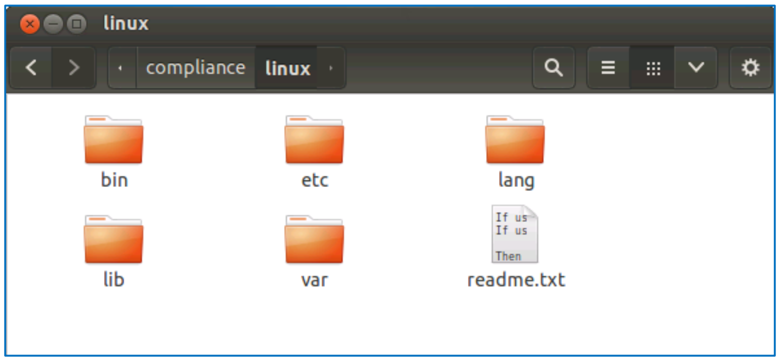
After getting the package, extract it to a desired directory (for example, extract the package to a directory: "**compliance/win**" for Windows, "**compliance/linux**" for Linux, and "**compliance/mac**" for Mac), and then you can see the resource files for Compliance are as follows:

For **Windows**:



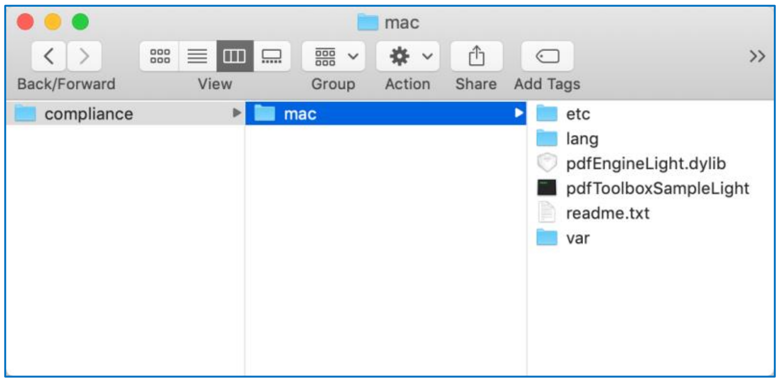
etc
lang
lib
var
readme.txt

For **Linux**:



bin
etc
lang
lib
var
readme.txt

For **Mac**:



compliance
mac
etc
lang
pdfEngineLight.dylib
pdfToolboxSampleLight
readme.txt
var

3.35.3 How to run the compliance demo

Foxit PDF SDK provides a **compliance** demo located in the "\examples\simple_demo\compliance" folder to show you how to use Foxit PDF SDK to verify whether a PDF is compliance with PDF/A standard, and convert a PDF to be compliance with PDF/A standard, as well as convert PDF versions.

3.35.3.1 Build a compliance resource directory

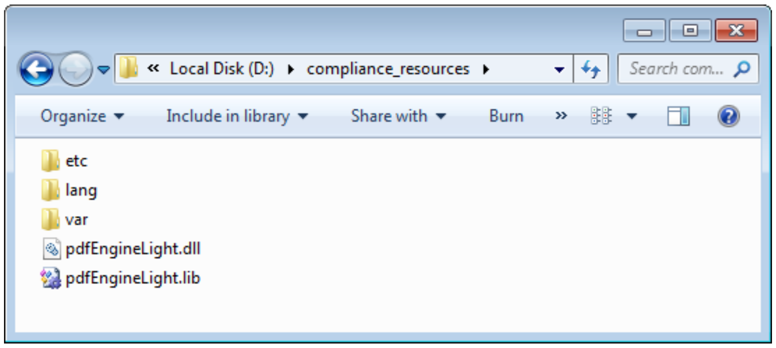
Before running the **compliance** demo, you should first build a compliance resource directory, and then pass the directory to Foxit PDF SDK API **ComplianceEngine::Initialize** to initialize compliance engine.

Windows

To build a compliance resource directory on Windows, please follow the steps below:

1) Create a new folder to add the resources. For example, "D:/compliance_resources".
2) Copy the whole folders of "**ect**", "**lang**", "**var**" under the "compliance/win" to "D:/compliance_resources".
3) Add the appropriate library resource based on the platform architecture.
   - For **win32**, copy **all the files** under "compliance/win/lib/x86" folder to "D:/compliance_resources".
   - For **win64**, copy **all the files** under "compliance/win/lib/x64" folder to "D:/compliance_resources".

For example, use **win32** platform architecture, then the compliance resource directory should be as follows:



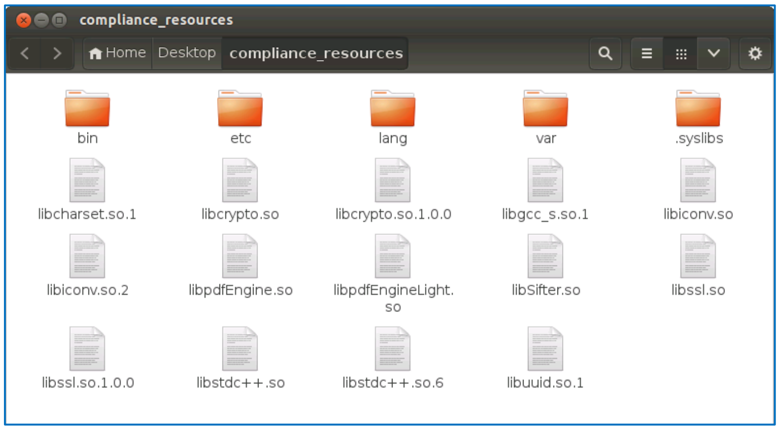
Linux

To build a compliance resource directory on Linux, please follow the steps below:

1) Create a new folder to add the resources. For example, "/root/Desktop/compliance_resources".
2) Copy the whole folders of "**bin**", "**ect**", "**lang**", "**var**" under the "compliance/linux" to "/root/Desktop/compliance_resources".
3) Add the appropriate library resource based on the platform architecture.

- For **linux32**, copy **all the files** under "compliance/linux/lib/x86" folder to "/root/Desktop/compliance_resources".
- For **linux64**, copy **all the files** under "compliance/linux/lib/x64" folder to "/root/Desktop/compliance_resources".

For example, use **linux32** platform architecture, then the compliance resource directory should be as follows:



***Note**: For Linux platform, you should put the compliance resource directory into the search path for system shared library before running the demo, otherwise **ComplianceEngine::Initialize** will fail.*

For example, you can use the command (export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}: /root/Desktop/compliance_resources) to temporarily add the compliance resource directory to LD_LIBRARY_PATH.

Mac

For Mac platform, you can directly use the "compliance/mac" resource folder as the compliance resource directory.

3.35.3.2 Configure the demo

After building the compliance resource directory, configure the demo in the "\examples\simple_demo\compliance\compliance.cpp" file.

This section takes **Windows** as an example to show you how to configure the demo in the "compliance.cpp" file. For Linux and Mac platforms, do the same configuration with Windows.

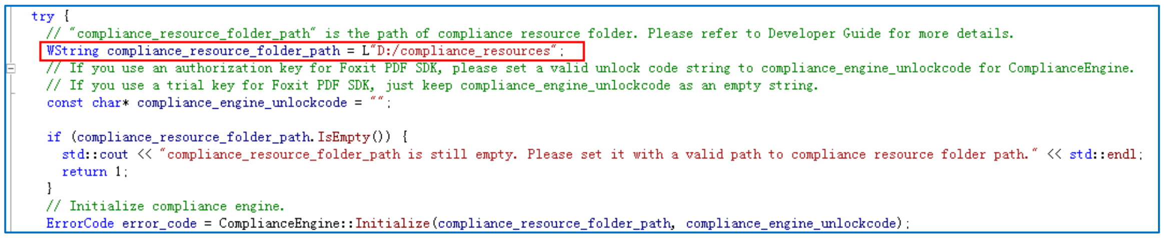
To load the **compliance** demo (for Windows) in Visual Studio, please choose one of the following ways:

1) Load the visual studio solution files "simple_demo_vs2010.sln" or "simple_demo_vs2015.sln" or "simple_demo_vs2017.sln" (depending on your Visual Studio version) in the "\examples\simple_demo" folder. Right-click the compliance demo, choose **Set as StartUp Project**.

2) Load the "compliance_vs2010.vcxproj" or "compliance_vs2015.vcxproj" or "compliance_vs2017.vcxproj" (depending on your Visual Studio version) in the "\examples\simple_demo\compliance" folder.

Specify the compliance resource directory

In the "compliance.cpp" file, add the compliance resource directory as follows, which will be used to initialize the compliance engine.



```
try {
  // "compliance_resource_folder_path" is the path of compliance resource folder. Please refer to Developer Guide for more details.
  WString compliance_resource_folder_path = L"D:/compliance_resources";
  // If you use an authorization key for Foxit PDF SDK, please set a valid unlock code string to compliance_engine_unlockcode for ComplianceEngine.
  // If you use a trial key for Foxit PDF SDK, just keep compliance_engine_unlockcode as an empty string.
  const char* compliance_engine_unlockcode = "";

  if (compliance_resource_folder_path.IsEmpty()) {
    std::cout << "compliance_resource_folder_path is still empty. Please set it with a valid path to compliance resource folder path." << std::endl;
    return 1;
  }
  // Initialize compliance engine.
  ErrorCode error_code = ComplianceEngine::Initialize(compliance_resource_folder_path, compliance_engine_unlockcode);
```

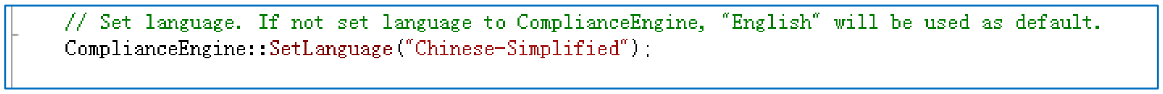
***Note**: If you have purchased an authorization license key (includes 'Compliance' module permission), Foxit sales team will send you an extra unlock code for initializing compliance engine.*

(Optional) Set language for compliance engine

ComplianceEngine::SetLanguage function is used to set language for compliance engine. The default language is "English", and the supported languages are as follows:

"Czech", "Danish", "Dutch", "English", "French", "Finnish", "German", "Italian", "Norwegian", "Polish", "Portuguese", "Spanish", "Swedish", "Chinese-Simplified", "Chinese-Traditional", "Japanese", "Korean".

For example, uncomment the **ComplianceEngine::SetLanguage** method, and set the language to "Chinese-Simplified".



```
    // Set language. If not set language to ComplianceEngine, "English" will be used as default.
    ComplianceEngine::SetLanguage("Chinese-Simplified");
```

(Optional) Set a temp folder for compliance engine

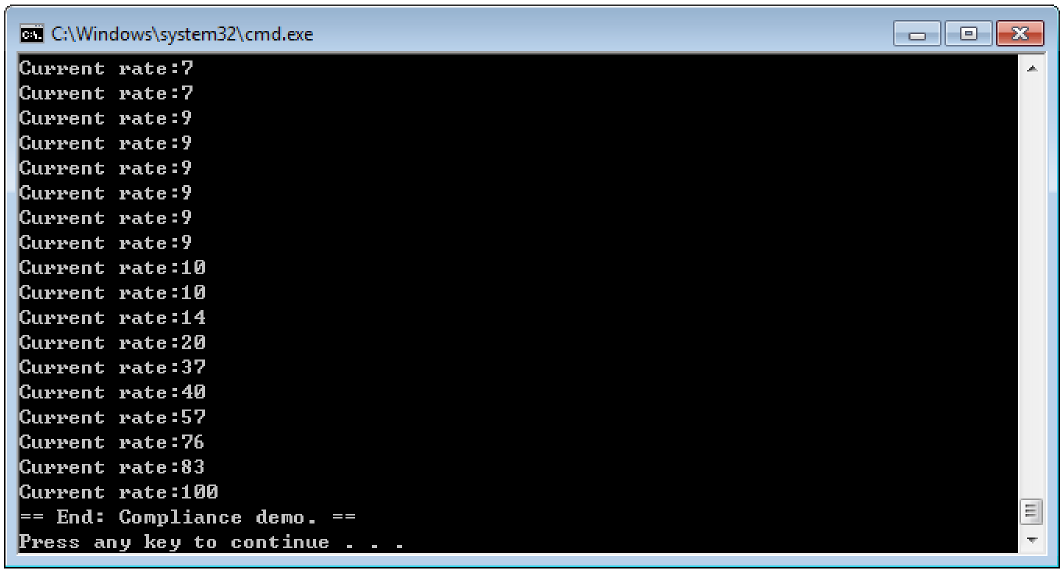
ComplianceEngine::SetTempFolderPath function is used to set a temp folder to store several files for proper processing (e.g verifying or converting). If no custom temp folder is set by this function, the default temp folder in system will be used.

For example, uncomment the **ComplianceEngine::SetTempFolderPath** method, and set the path to "D:/compliance_temp" (should be a valid path).

```
    // Set custom temp folder path for ComplianceEngine.
    ComplianceEngine::SetTempFolderPath(L"D:/compliance_temp");
```

3.35.3.3 Run the demo

Once you run the demo successfully, the console will print the following by default:



```
Current rate:7
Current rate:7
Current rate:9
Current rate:9
Current rate:9
Current rate:9
Current rate:9
Current rate:9
Current rate:10
Current rate:10
Current rate:14
Current rate:20
Current rate:37
Current rate:40
Current rate:57
Current rate:76
Current rate:83
Current rate:100
== End: Compliance demo. ==
Press any key to continue . . .
```

The demo will

- verify whether the PDF ("\examples\simple_demo\input_files\AboutFoxit.pdf") is compliance with PDF/A-1a standard, and convert the PDF to be compliance with PDF/A-1a standard.
- convert PDF file ("\examples\simple_demo\input_files\AF_ImageXObject_FormXObject.pdf") to PDF-1.4 and PDF-1.7.

The output files are located in "\examples\simple_demo\output_files\compliance" folder.

3.36 Optimization

Optimization feature can reduce the size of PDF files to save disk space and make files easier to send and store, through compressing images, deleting redundant date, discarding useless user data and so on. From version 7.0, optimization module provides functions to compress the color/grayscale/monochrome images in PDF files to reduce the size of the PDF files.

***Note**: To use the Optimization feature, please make sure the license key has the permission of the 'Optimization' module.*

Example:

3.36.1 How to optimize PDF files by compressing the color/grayscale/monochrome images

```
#include "include/common/fs_common.h"
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/fs_pdfpage.h"
#include "include/addon/optimization/fs_optimization.h"

using namespace std;
using namespace foxit;
using namespace foxit::common;
using namespace pdf;
using namespace foxit::addon;

PDFDoc doc("input_pdf_file");
ErrorCode error_code = doc.Load();
if (error_code != foxit::e_ErrSuccess) {
		printf("Error: %d\n", error_code);
		return 1;
}
Optimization_Pause pause(0,true);
addon::optimization::OptimizerSettings settings;
common::Progressive progressive =  addon::optimization::Optimizer::Optimize(doc,settings,&pause);
cout << "Optimized Start." << endl;
Progressive::State progress_state = Progressive::e_ToBeContinued;
while (Progressive::e_ToBeContinued == progress_state) {
		progress_state = progressive.Continue();
		int percent = progressive.GetRateOfProgress();
		String res_string;
		res_string.Format("Optimize progress percent: %d %",percent);
		std::cout<<res_string<<std::endl;
}
if(Progressive::e_Finished ==  progress_state)
{
		doc.SaveAs(L"ImageCompression_Optimized.pdf",
foxit::pdf::PDFDoc::e_SaveFlagRemoveRedundantObjects);

}
cout << "Optimized Finish." << endl;
```

3.37 HTML to PDF Conversion

For some large HTML files or a webpage which contain(s) many contents, it is not convenient to print or archive them directly. Foxit PDF SDK provides APIs to convert the online webpage or local HTML files like invoices or reports into PDF file(s), which makes them easier to print or archive. In the process of conversion from HTML to PDF, Foxit PDF SDK also supports to create and add PDF Tags based on the organizational structure of HTML.

This section will provide instructions on how to set up your environment for running the 'html2pdf' demo.

3.37.1 System requirements

Platform: Windows, Mac

Programming Language: C++, Java, C#, Objective-C

License Key requirement: 'Conversion' module permission in the license key

SDK Version: Foxit PDF SDK 7.0 or higher

3.37.2 HTML to PDF engine files

Please contact Foxit support team or sales team to get the HTML to PDF engine files package.

After getting the package, extract it to a desired directory (for example, extract the package to a directory: "**D:/htmltopdf/win**" for Windows, and "**htmltopdf/mac**" for Mac).

3.37.3 How to run the html2pdf demo

Foxit PDF SDK provides a html2pdf demo located in the "\examples\simple_demo\html2pdf" folder to show you how to use Foxit PDF SDK to convert from html to PDF.

3.37.3.1 Configure the demo

For html2pdf demo, you can configure the demo in the "\examples\simple_demo\html2pdf\html2pdf.cpp" file, or you can configure the demo with parameters directly in a command prompt or a terminal. Following will configure the demo in "html2pdf.cpp" file on Windows for example. For Mac platform, do the same configuration with Windows.

To load the html2pdf demo (for Windows) in Visual Studio, please choose one of the following ways:

1) Load the visual studio solution files "simple_demo_vs2010.sln" or "simple_demo_vs2015.sln" or "simple_demo_vs2017.sln" (depending on your Visual Studio version) in the "\examples\simple_demo" folder. Right-click the html2pdf demo, choose **Set as StartUp Project**.

2) Load the "html2pdf_vs2010.vcxproj" or "html2pdf_vs2015.vcxproj" or "html2pdf_vs2017.vcxproj" (depending on your Visual Studio version) in the "\examples\simple_demo\html2pdf" folder.

Specify the html2pdf engine directory

In the "html2pdf.cpp" file, add the path of the engine file "fxhtml2pdf.exe" as follows, which will be used to convert html files to PDF files.

```
// "engine_path" is the path of the engine file "fxhtml2pdf" which is used to converting html to pdf. Please refer to Developer Guide for more details.
WString engine_path = L"D:/htmltopdf/win/fxhtml2pdf"; // or engine_path = L"D:/htmltopdf/win/fxhtml2pdf.exe".
```

(Optional) Specify cookies file path

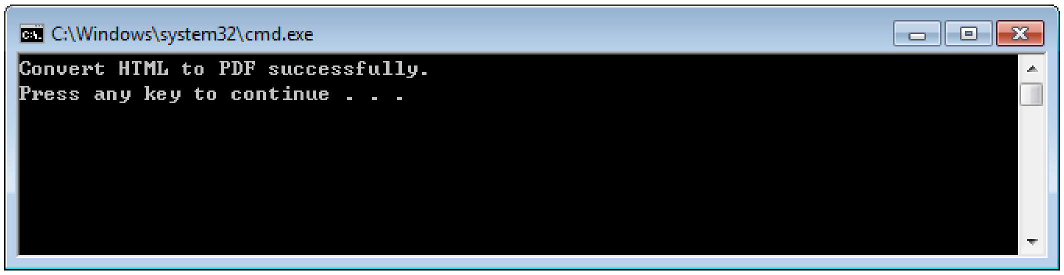
Add the path of the cookies file exported from the web pages that you want to convert. For example,

```
// "cookies_path" is the path of the cookies file exported from the web pages that you want to convert. Please refer to Developer Guide for more details.
WString cookies_path = L"D:/cookies.txt";
```

3.37.3.2 Run the demo

Run the demo without parameters

Once you run the demo successfully, the console will print the following by default:

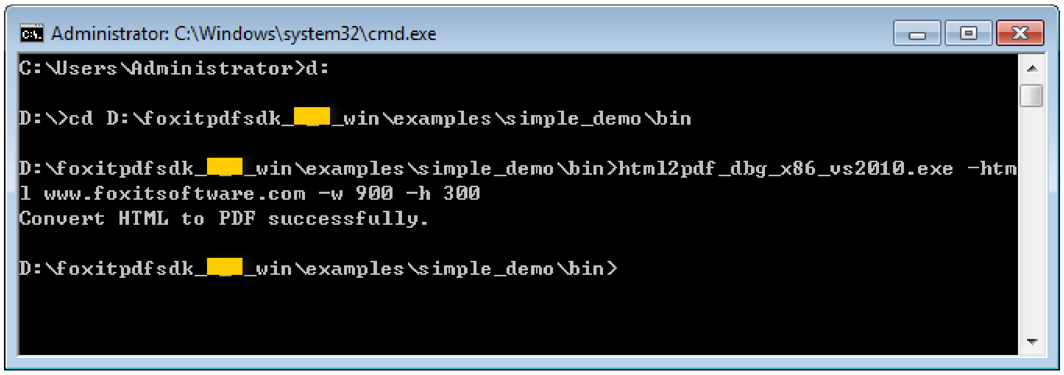


```
C:\Windows\system32\cmd.exe
Convert HTML to PDF successfully.
Press any key to continue . . .
```

Run the demo with parameters

After building the demo successfully, open a command prompt, navigate to "\examples\simple_demo\bin", type "html2pdf_dbg_x86_vs2010.exe --help" for example to see how to use the parameters to execute the program.

For example, convert the URL web page "www.foxitsoftware.com" into a PDF with setting the page width to 900 points and the page height to 300 points:



```
Administrator: C:\Windows\system32\cmd.exe
C:\Users\Administrator>d:

D:\>cd D:\foxitpdfsdk_[redacted]_win\examples\simple_demo\bin

D:\foxitpdfsdk_[redacted]_win\examples\simple_demo\bin>html2pdf_dbg_x86_vs2010.exe -htm
l www.foxitsoftware.com -w 900 -h 300
Convert HTML to PDF successfully.

D:\foxitpdfsdk_[redacted]_win\examples\simple_demo\bin>
```

The output file is located in "\examples\simple_demo\output_files\html2pdf" folder.

Parameters Description

Basic Syntax:

html2pdf *<-html <the url or html path>> <-o <output pdf path>> <-engine <htmltopdf engine path>>*
[-w <page width>] [-h <page height>] [-ml <margin left>] [-mr <margin right>]
[-mt <margin top>] [-mb <margin bottom>] [-r <page rotate degree>] [-mode <page mode>]
[-scale <whether scale page>] [-link <whether convert link>] [-tag <whether generate tag>]
[-cookies <cookies file path>] [-timeout <timeout>]

html2pdf *--help*

Note:

- <> required
- [ ] optional

Parameters	Description
--help	The usage description of the parameters.
-html	The url or html file path. For examples '-html www.foxitsoftware.com'.
-o	The path of the output PDF file.
-engine	The path of the engine file "fxhtml2pdf.exe".
-w	The page width of the output PDF file in points.
-h	The page height of the output PDF file in points.
-r	The page rotate for the output PDF file. • 0 : 0 degree. • 1 : 90 degree. • 2 : 180 degree. • 3 : 270 degree.
-ml	The left margin of the pages for the output PDF file.
-mr	The right margin of the pages for the output PDF file.
-mt	The top margin of the pages for the output PDF file.
-mb	The bottom margin of the pages for the output PDF file.
-mode	The page mode for the output PDF file. • 0 : single page mode. • 1 : multiple pages mode.
-scale	Whether to scale pages. • 'yes' : scale pages. • 'no' : No need to scale pages.
-link	Whether to convert links.

Parameters	Description
	• 'yes' : convert links. • 'no' : No need to convert links.
-tag	Whether to generate tag. • 'yes' : generate tag. • 'no' : No need to generate tag.
-cookies	The path of the cookies file exported from a URL that you want to convert.
-timeout	The timeout of loading webpages.

3.38 Word/Excel to PDF Conversion

From version 7.3, Foxit PDF SDK provides APIs to convert Microsoft Office documents (Word and Excel) into professional-quality PDF files on Windows platform.

For using this feature, please note that:

- Make sure that Microsoft Office 2007 version or higher is already installed on your Windows system.
- Before converting Excel to PDF, make sure that the default Microsoft virtual printerk is already set on your Windows system.

3.38.1 System requirements

Platform: Windows

Programming Language: C++, Java, C#

License Key requirement: 'Conversion' module permission in the license key

SDK Version: Foxit PDF SDK 7.3

Example:

3.38.2 How to convert Word to PDF

```
#include "include/addon/conversion/fs_convert.h"
...
// Make sure that SDK has already been initialized successfully.

WString word_file_path = L"test.doc";
WString output_path = L"saved.pdf";

// Use default Word2PDFSettingData values.
foxit::addon::conversion::Word2PDFSettingData word_convert_setting_data;
foxit::addon::conversion::Convert::FromWord(word_file_path, L"", output_path, word_convert_setting_data);
```

3.38.3 How to convert Excel to PDF

```
#include "include/addon/conversion/fs_convert.h"
...
// Make sure that SDK has already been initialized successfully.

WString excel_file_path = L"test.xls";
WString output_path = L"saved.pdf";

// Use default Excel2PDFSettingData values.
foxit::addon::conversion::Excel2PDFSettingData excel_convert_setting_data;
foxit::addon::conversion::Convert::FromExcel(excel_file_path, L"", output_path, excel_convert_setting_data);
```

FAQ

1. How to fix the "'xcopy' exited with code 9009" error when building demos in Visual Studio?

When building demos in Visual Studio, if you encounter the error "'xcopy' exited with code 9009" as follows:

```
'xcopy ..\..\..\..\..\..\lib\gsdk_sn.txt ..\..\..\ /y > null
xcopy ..\..\..\..\..\..\lib\gsdk_key.txt ..\..\..\ /y > null
xcopy ..\..\..\..\..\..\lib\$(PlatformName)_vc10\fsdk.dll ..\..\..\ /y > null
xcopy ..\..\..\..\..\..\lib\$(PlatformName)_vc10\fsdk_dotnet.dll ..\..\..\ /y > null' exited with code 9009
```

Please check the following points:

1) Check whether the **xcopy.exe** is in the *"%SystemRoot%\System32"* directory, if not, copy one from another machine.

2) Check whether the system **PATH** environment variables have been set correctly. It should contain *"%SystemRoot%\System32;%SystemRoot%;"*, if the environment variables for **xcopy** is right, but it still reports the error, please put the path of **xcopy** in front of others. Maybe some other environment variables have spell mistakes, so that cause the subsequent environment variables are invalid. Please check it.

After checking, open a command prompt, type xcopy command, if it can be recognized, close Visual Studio, and restart the demos. The error should be fixed.

2. How do I get text objects in a specified position of a PDF and change the contents of the text objects?

To get text objects in a specified position of a PDF and change the contents of the text objects using Foxit PDF SDK, you can follow the steps below:

1) Open a PDF file.

2) Load PDF pages and get the page objects.

3) Use **PDFPage::GetGraphicsObjectAtPoint** to get the text object at a certain position. Note: use the page object to get rectangle to see the position of the text object.

4) Change the contents of the text objects and save the PDF document.

Following is the sample code:

```
#include "include/common/fs_common.h"
```

```
#include "include/pdf/fs_pdfdoc.h"
#include "include/pdf/graphics/fs_pdfgraphicsobject.h"

using namespace std;
using namespace foxit;
using namespace foxit::common;
using foxit::common::Library;
using namespace pdf;
using namespace graphics;
...

bool ChangeTextObjectContent()
{
    try {
            WString input_file = input_path + L"AboutFoxit.pdf";
            PDFDoc doc(input_file);
            ErrorCode error_code = doc.Load();
            if (error_code != foxit::e_ErrSuccess) {
                    printf("The Doc [%s] Error: %d\n", (const char*)String::FromUnicode(input_file),
error_code);
                    return false;
            }
            // Get original shading objects from the first PDF page.
            PDFPage original_page = doc.GetPage(0);
            original_page.StartParse(PDFPage::e_ParsePageNormal, NULL, false);
            foxit::PointF pointf;
            pointf.x = 92;
            pointf.y = 762;
            GraphicsObjectArray arr = original_page.GetGraphicsObjectsAtPoint (pointf, 10,
GraphicsObject::e_TypeText );
            for(int i = 0; i<arr.GetSize(); i++)
            {
                    GraphicsObject* graphobj = arr.GetAt(i);
                    TextObject * textobj = graphobj->GetTextObject();
                    textobj->SetText(L"Foxit Test");
            }
            original_page.GenerateContent();
            WString output_directory = output_path + L"graphics_objects/";
            WString output_file = output_directory + L"After_revise.pdf";
            doc.SaveAs(output_file, PDFDoc::e_SaveFlagNormal);
    }catch (const Exception& e) {
            cout << e.GetMessage() << endl;
            return false;
    }
    return true;
}
```

3. Can I change the DPI of an embedded TIFF image?

No, you cannot change it. The DPI of the images in PDF files is static, so if the images already exist, Foxit PDF SDK does not have functions to change its DPI.

The solution is that you can use third-party library to change the DPI of an image, and then add it to the PDF file.

***Note**: Foxit PDF SDK provides a function "Image::SetDPIs" which can set the DPI property of an image object. However, it only supports the images that are created by Foxit PDF SDK or created by function "Image::AddFrame", and it does not support the image formats of JPX, GIF and TIF.*

APPENDIX

Supported JavaScript List

Objects' property or method

Object	Properites/Method Names	Minimum Supported SDK Version
annotation properties	alignment	V7.0
	author	V7.0
	contents	V7.0
	creationDate	V7.0
	fillColor	V7.0
	hidden	V7.0
	modDate	V7.0
	name	V7.0
	opacity	V7.0
	page	V7.0
	readOnly	V7.0
	rect	V7.0
	richContents	V7.1
	rotate	V7.0
	strokeColor	V7.0
	textSize	V7.0
	type	V7.0
annotation method	destroy	V7.0
app properties	activeDocs	V4.0
	calculate	V4.0
	formsVersion	V4.0
	fs	V4.0
	fullscreen	V4.0
	language	V4.2
	platform	V4.0
	runtimeHighlight	V4.0
	viewerType	V4.0
	viewerVariation	V4.0
	viewerVersion	V4.0
app methods	alert	V4.0
	beep	V4.0
	browseForDoc	V4.0
	clearInterval	V4.0
	clearTimeOut	V4.0

Object	Properites/Method Names	Minimum Supported SDK Version
	launchURL	V4.0
	mailMsg	V4.0
	response	V4.0
	setInterval	V4.0
	setTimeOut	V4.0
	popUpMenu	V4.0
color properties	black	V4.0
	blue	V4.0
	cyan	V4.0
	dkGray	V4.0
	gray	V4.0
	green	V4.0
	ltGray	V4.0
	magenta	V4.0
	red	V4.0
	transparent	V4.0
	white	V4.0
	yellow	V4.0
color methods	convert	V4.0
	equal	V4.0
document properties	author	V4.0
	baseURL	V4.0
	bookmarkRoot	V7.0
	calculate	V4.0
	Collab	V4.0
	creationDate	V4.0
	creator	V4.0
	delay	V4.0
	dirty	V4.0
	documentFileName	V4.0
	external	V4.0
	filesize	V4.0
	icons	V4.0
	info	V4.0
	keywords	V4.0
	modDate	V4.0
	numFields	V4.0
	numPages	V4.0
	pageNum	V4.0

Object	Properites/Method Names	Minimum Supported SDK Version
	path	V4.0
	producer	V4.0
	subject	V4.0
	title	V4.0
document methods	addAnnot	V7.0
	addField	V4.0
	addIcon	V4.0
	calculateNow	V4.0
	createDataObject	V6.2
	deletePages	V4.0
	exportAsFDF	V4.0
	flattenPages	V7.1
	getAnnot	V7.0
	getAnnots	V7.0
	getField	V4.0
	getIcon	V4.0
	getNthFieldName	V4.0
	getOCGs	V4.0
	getPageBox	V4.0
	getPageNthWord	V4.0
	getPageNthWordQuads	V4.0
	getPageNumWords	V4.0
	getPageRotation	V7.0
	getPrintParams	V4.0
	getURL	V4.0
	importAnFDF	V4.0
	insertPages	V6.2
	mailForm	V4.0
	print	V4.0
	removeField	V4.0
	replacePages	V6.2
	resetForm	V4.0
	submitForm	V4.0
	mailDoc	V4.0
event properties	change	V4.0
	changeEx	V4.0
	commitKey	V4.0
	fieldFull	V4.0
	keyDown	V4.0

Object	Properites/Method Names	Minimum Supported SDK Version
	modifier	V4.0
	name	V4.0
	rc	V4.0
	selEnd	V4.0
	selStart	V4.0
	shift	V4.0
	source	V4.0
	target	V4.0
	targetName	V4.0
	type	V4.0
	value	V4.0
	willCommit	V4.0
event methods	-	-
field properties	alignment	V4.0
	borderStyle	V4.0
	buttonAlignX	V4.0
	buttonAlignY	V4.0
	buttonFitBounds	V4.0
	buttonPosition	V4.0
	buttonScaleHow	V4.0
	buttonScaleWhen	V4.0
	calcOrderIndex	V4.0
	charLimit	V4.0
	comb	V4.0
	commitOnSelChange	V4.0
	currentValueIndices	V4.0
	defaultValue	V4.0
	doNotScroll	V4.0
	doNotSpellCheck	V4.0
	delay	V4.0
	display	V4.0
	doc	V4.0
	editable	V4.0
	exportValues	V4.0
	hidden	V4.0
	fileSelect	V4.0
	fillColor	V4.0
	lineWidth	V4.0
	highlight	V4.0

Object	Properites/Method Names	Minimum Supported SDK Version
	multiline	V4.0
	multipleSelection	V4.0
	name	V4.0
	numItems	V4.0
	page	V4.0
	password	V4.0
	print	V4.0
	radiosInUnison	V4.0
	readonly	V4.0
	rect	V4.0
	required	V4.0
	richText	V4.0
	rotation	V4.0
	strokeColor	V4.0
	style	V4.0
	textColor	V4.0
	textFont	V4.0
	textSize	V4.0
	type	V4.0
	userName	V4.0
	value	V4.0
	valueAsString	V4.0
field methods	browseForFileToSubmit	V4.0
	buttonGetCaption	V4.0
	buttonGetIcon	V4.0
	buttonSetCaption	V4.0
	buttonSetIcon	V4.0
	checkThisBox	V4.0
	clearItems	V4.0
	defaultIsChecked	V4.0
	deleteItemAt	V4.0
	getArray	V4.0
	getItemAt	V4.0
	insertItemAt	V4.0
	isBoxChecked	V4.0
	isDefaultChecked	V4.0
	setAction	V4.0
	setFocus	V4.0
	setItems	V4.0

Object	Properites/Method Names	Minimum Supported SDK Version
global methods	setPersistent	V4.0
Icon properties	name	V4.0
util methods	printd	V4.0
	printf	V4.0
	printx	V4.0
	scand	V4.0
identity properties	loginName	V4.2
	Name	V4.2
	corporation	V4.2
	email	V4.2
collab properties	user	V6.2
ocg properties	name	V6.2
ocg methods	setAction	V6.2

Global methods

Method Names	Minimum Supported SDK Version
AFNumber_Format	V4.0
AFNumber_Keystroke	V4.0
AFPercent_Format	V4.0
AFPercent_Keystroke	V4.0
AFDate_FormatEx	V4.0
AFDate_KeystrokeEx	V4.0
AFDate_Format	V4.0
AFDate_Keystroke	V4.0
AFTime_FormatEx	V4.0
AFTime_KeystrokeEx	V4.0
AFTime_Format	V4.0
AFTime_Keystroke	V4.0
AFSpecial_Format	V4.0
AFSpecial_Keystroke	V4.0
AFSpecial_KeystrokeEx	V4.0
AFSimple	V4.0
AFMakeNumber	V4.0
AFSimple_Calculate	V4.0
AFRange_Validate	V4.0
AFMergeChange	V4.0
AFParseDateEx	V4.0
AFExtractNums	V4.0

REFERENCES

[1] PDF reference 1.7

http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=51502

[2] PDF reference 2.0

https://www.iso.org/standard/63534.html

[3] Foxit PDF SDK API reference

sdk_folder/doc/Foxit PDF SDK API Reference.html

Note: sdk_folder is the directory of unzipped package.

SUPPORT

Foxit support home link:

http://www.foxitsoftware.com/support/

Sales contact phone number:

Phone: 1-866-680-3668

Email: sales@foxitsoftware.com

Support & General contact:

Phone: 1-866-MYFOXIT or 1-866-693-6948

Email: support@foxitsoftware.com