Quick Start 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 an Amazon-invested 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 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 30-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 to 10.12	

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 6.2.1, so it shows 6_2_1.

2.2.1 What is in the package

Download Foxit PDF SDK zip for Windows package and extract it to a new directory "foxitpdfsdk_6_2_1_win", which is shown in Figure 2-1. The release package contains the following folders:

docs: API references, quick start guide

include: header files for foxit pdf sdk API

lib: libraries and license files

samples: sample projects and demos

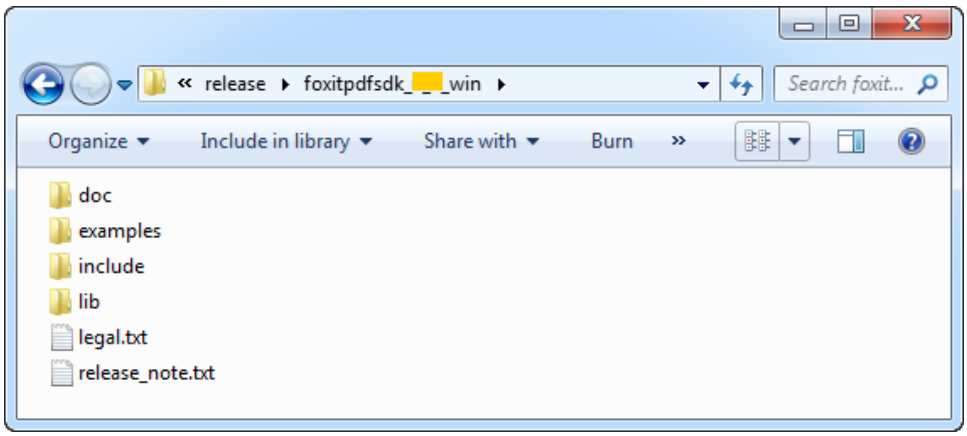

Figure 2-1

In the "examples" folder, there are two types of demos. "examples\simple_demo" contains more than 20 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 PDF SDK APIs to complete their applications.

To run a demo in Visual Studio (expect connectedpdf, security and signature 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 (See Figure 2-2). And the names of the executable files depend on the build configurations.

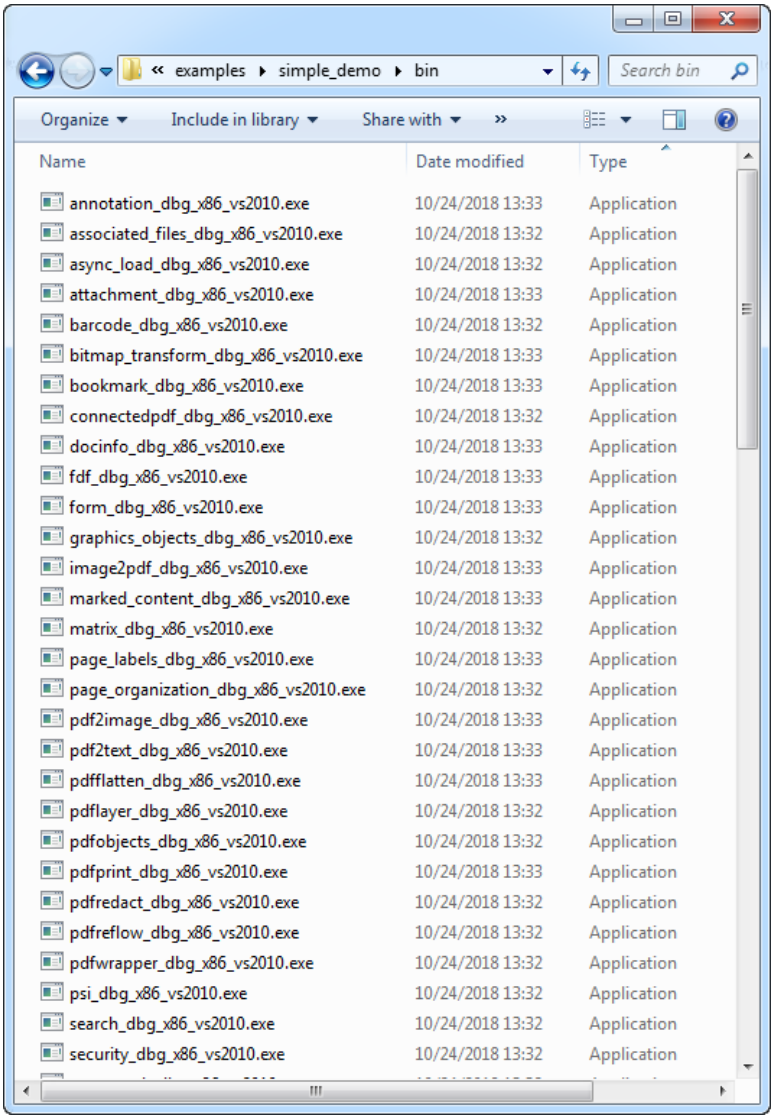

Figure 2-2

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 "example\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.*

ConnectedPDF demo

Before running **connectedpdf** demo, please make sure to configure the following settings in "connectedpdf.m" under "examples\simple_demo\connectedpdf" folder. Then run the demo according to the other demos.

```
String endpoint = "";        // Please write server address here.
String user_email = "";      // Please write user name here.
String password = "";        // Please write password here.
```

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.

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

2) 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".

3) 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 "libs" 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.*

View Demo

This view demo provides an example for developers to realize a PDF reader using 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 2-3

You can drag a PDF file to the above window (Figure 2-3) 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 2-4.

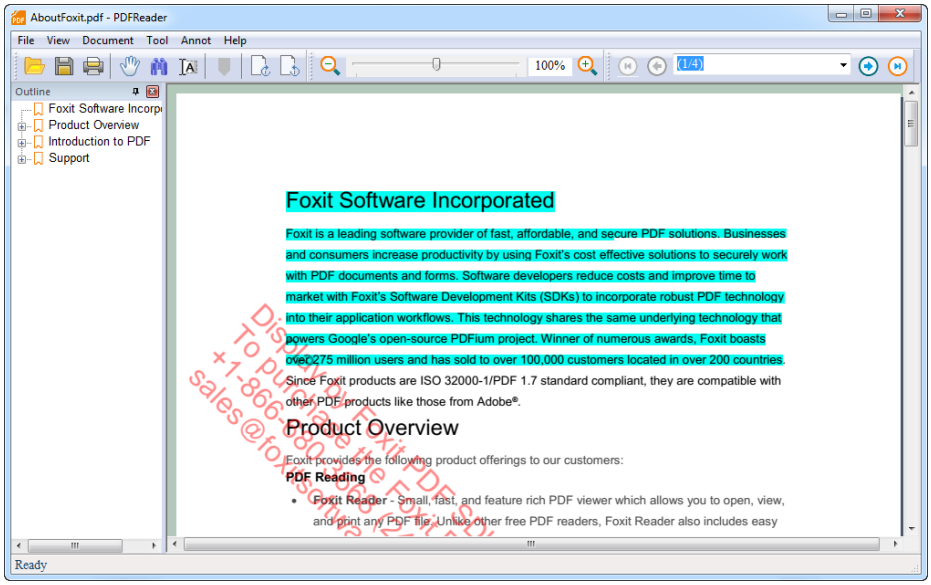

Figure 2-4

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_6_2_1_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 "libs" 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 "libs" 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 6.2.1, so it shows 6_2_1.

2.3.1 What is in this package

Download Foxit PDF SDK zip for Linux package and extract it to a new directory "foxitpdfsdk_6_2_1_linux".The structure of the release package is shown in Figure 2-5. This package contains the following folders:

docs: API references, quick start guide

include: header files for foxit pdf sdk API

lib: libraries and license files

samples: sample projects and demos

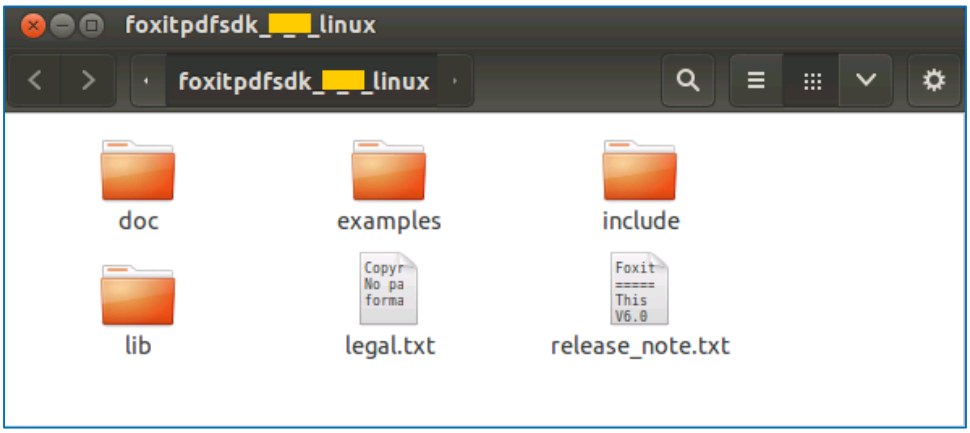

Figure 2-5

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

2.3.2 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
Linux	GCC	4.8 and later

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

1) Open a terminal window, navigate to "foxitpdfsdk_6_2_1_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 "example/simple_demo/output_files/".

ConnectedPDF demo

Before running **connectedpdf** demo, please make sure to configure the following settings in "connectedpdf.m" under "examples\simple_demo\connectedpdf" folder. Then run the demo according to the other demos.

```
String endpoint = "";        // Please write server address here.
String user_email = "";      // Please write user name here.
String password = "";        // Please write password here.
```

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 "libs" 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.*

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_6_2_1_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 6.2.1, so it shows 6_2_1.

2.4.1 What is in this package

Download Foxit PDF SDK zip for Mac package and extract it to a new directory "foxitpdfsdk_6_2_1_mac". The structure of the release package is shown in Figure 2-5. This package contains the following folders:

docs: API references, quick start guide

include: header files for foxit pdf sdk API

lib: libraries and license files

samples: sample projects and demos

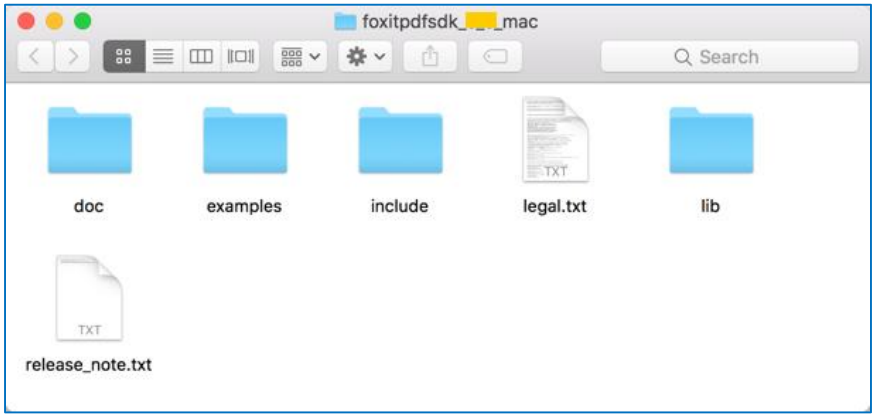

Figure 2-6

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 (expect connectedpdf, security and signature demos which will be introduced later), please follow the steps:

1) Open a terminal window, navigate to "foxitpdfsdk_6_2_1_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 "example/simple_demo/output_files/".

ConnectedPDF demo

Before running **connectedpdf** demo, please make sure to configure the following settings in "connectedpdf.m" under "examples\simple_demo\connectedpdf" folder. Then run the demo according to the other demos.

```
String endpoint = "";        // Please write server address here.
String user_email = "";      // Please write user name here.
String password = "";        // Please write password here.
```

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 "libs" 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.*

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_6_2_1_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.

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