Quick Start Guide

Foxit PDF SDK

For .NET

Microsoft® Partner
Gold Independent Software Vendor (ISV)

©Foxit Software Incorporated. All rights reserved.

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 .Net

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 Windows platform with .Net framework.

Foxit PDF SDK for .NET is a Component for Windows which ships with simple-to-use APIs that can help .NET developers seamlessly integrate powerful PDF technology into their own projects based on .NET Framework 4.0 or higher. It offers the most common features in PDF SDK, such as PDF viewing, bookmark navigating, text selecting/copying/searching, annotations, and signature.

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 .NET SDK 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 on how to integrate Foxit PDF SDK into the projects based on .NET Framework 4.0 or higher. 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

Windows XP, Vista, 7, 8 and 10 (32-bit, 64-bit)

Windows Server 2003, 2008 and 2012 (32-bit and 64-bit)

The release package includes a 32 bit version and native 64 bit version DLL library for Windows 32/64.

Visual Studio 2010/2015/2017 installed with .NET Framework 4.0 or higher

Note: It only supports for Windows 8/10 classic style, but not for Store App or UWP app.

2.2 What is in the package

Download Foxit PDF SDK zip for .NET package and extract it to a new directory "foxitpdfsdk_6_1_win_dotnet", which is shown in Figure 2-1.

***NOTE**: the highlighted rectangle in the figure is just the version of Foxit PDF SDK. Here the SDK version is 6.1, so it shows 6_1 in the package name.*

The release package contains the following folders:

docs:	API references, quick start guide
lib:	libraries and license files
examples:	sample projects and demos

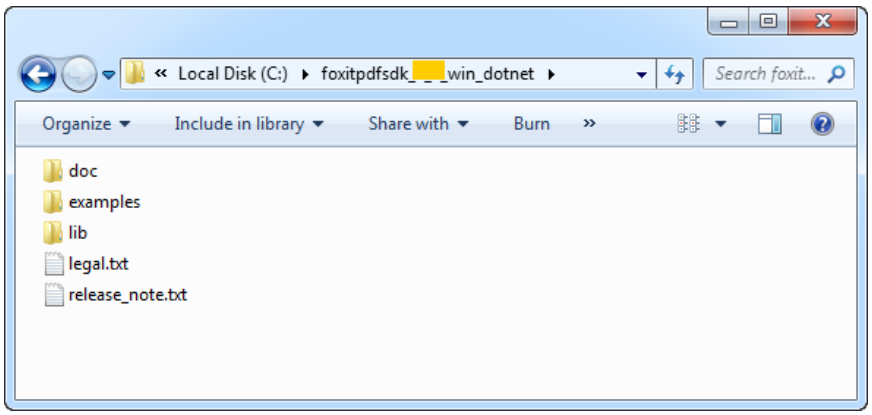

Figure 2-1

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.

2.3 How to run a demo

Simple Demo

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

To run a demo in Visual Studio 2010/2015/2017, you can follow the steps below:

1) Load the visual studio solution file "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 "*.csproj" 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.

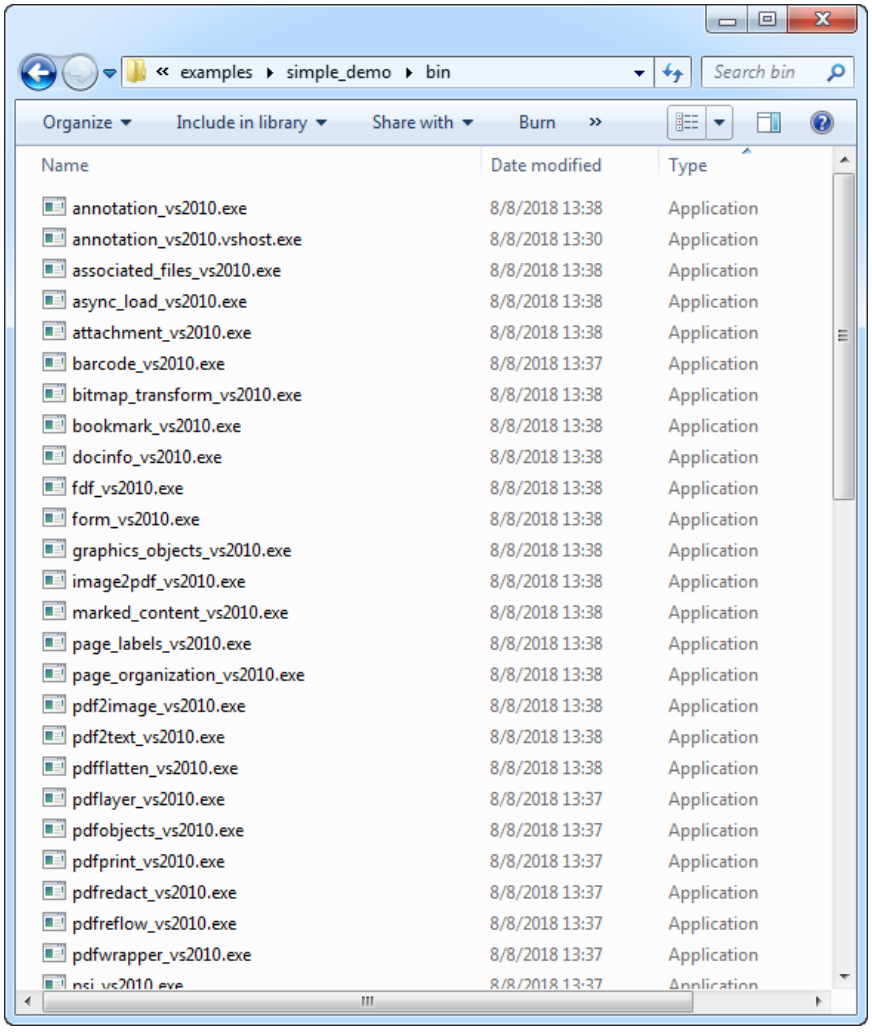

Figure 2-2

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

   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.

   In "examples\simple_demo\output_files**security**" folder, if you want to open the "certificate_encrypt.pdf" document, you should install the certificates "foxit.cer" and "foxit_all.pfx" found in "examples\simple_demo\input_files" folder at first. Please follow the steps below:

   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".

View Demo

This view demo provides an example for .NET 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" folder, and then click "Debug > Start Without Debugging" to run it. After the demo starts, you will see the following window as shown in Figure 2-3.

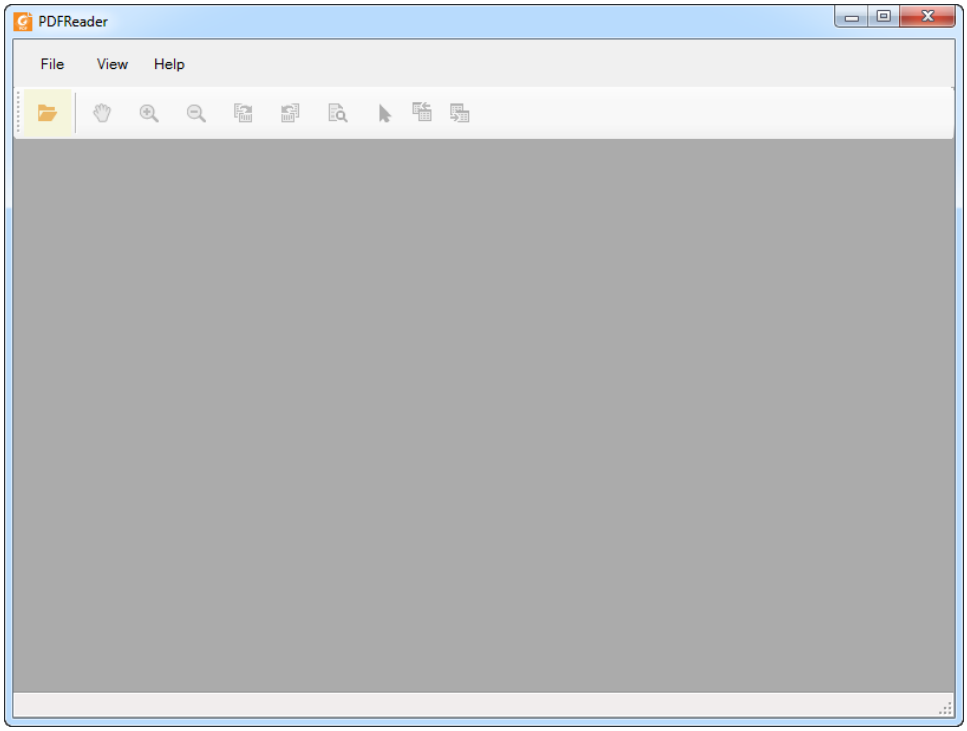

Figure 2-3

You can click on **File** > **Open** or click the directory icon to open a PDF file, for example, we open a PDF document named "AboutFoxit.pdf" found in "examples\simple_demo\input_files" folder.

This demo provides the features like rendering a PDF document, zooming, page rotation, text selection and search, and page turning. For example, click the Next Page button to view the next page, which is shown in Figure 2-4.

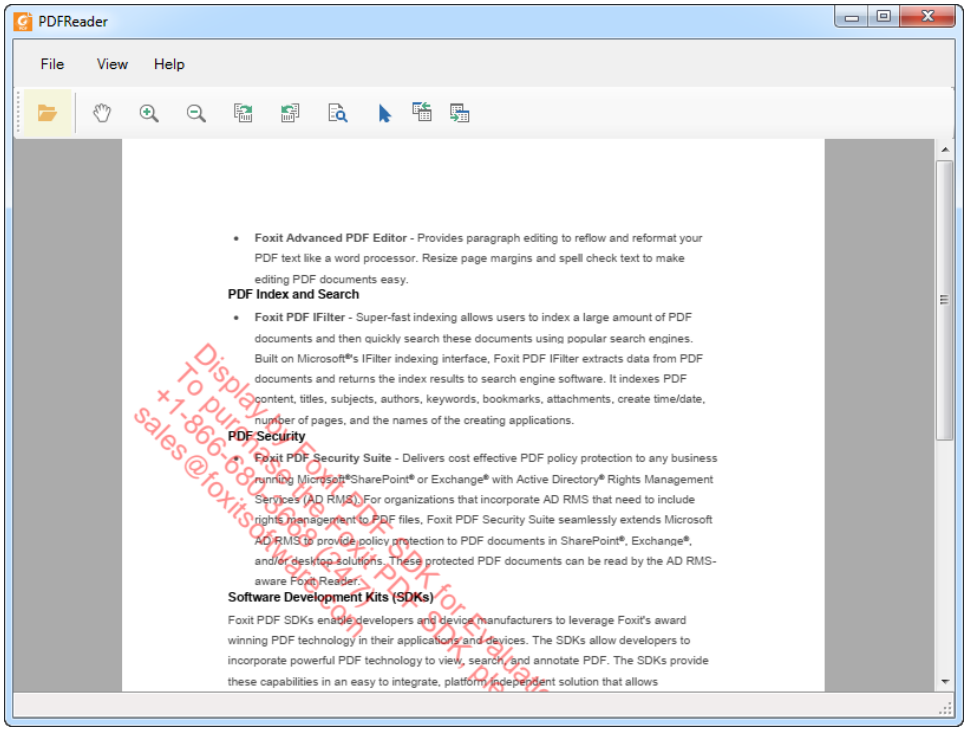

Figure 2-4

2.4 How to create a simple project

In this section, we will show you how to use Foxit PDF .NET SDK 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 C# Console Application named "test_dotnet".

2) Copy the "lib" folder from the "foxitpdfsdk_6_1_win_dotnet" folder to the project "test_ dotnet" folder.

3) Add Foxit PDF SDK dynamic library to **References**. In order to use Foxit PDF SDK APIs in the project, you must first add a reference to it.

   i. In Solution Explorer, right-click the "test_dotnet" project and click **Add Reference…**

   ii. In the **Add Reference** dialog, click **Browse** tab, navigate to the "test_dotnet\lib\x64_vc10" or "test_dotnet\lib\x86_vc10" folder depending on your build configurations, select **fsdk_dotnet.dll** dynamic library, and then click **OK**.

Note: *Please make sure that the "fsdk_dotnet.dll" architecture needs to match the platform target (Win32 or Win64) of the application.*

4) Add "System.Drawing" item to **References**.

   Right-click the "test_dotnet" project and click **Add Reference…** In the **Add Reference** dialog, click **.NET** tab and select **System.Drawing**, and then click **OK**.

5) Add "fsdk.dll" as source file to the project.

   Right-click the "test_dotnet" project and click **Add** > **Existing Item…**, navigate to the "test_dotnet\lib\x64_vc10" or "test_dotnet\lib\x86_vc10" folder depending on your build configurations, select **fsdk.dll dynamic library**, and click **Add**.

   Note*: Please make sure to set the property "Copy to Output Directory" of "fsdk.dll" to "Copy if newer". Otherwise, you should copy it to the same folder with the executable file manually before running the project.*

6) Add using statement to the beginning of the "Program.cs".

```
using System.Drawing;
using foxit;
using foxit.common;
using foxit.common.fxcrt;
using foxit.pdf;
```

7) Initialize 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.

```
string sn = " ";
string key = " ";
ErrorCode error_code = Library.Initialize(sn, key);
if (error_code != ErrorCode.e_ErrSuccess)
{
   return;
}
```

   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_dotnet\test_dotnet" folder.

```
PDFDoc doc = new PDFDoc("..\..\Sample.pdf");
error_code = doc.LoadW("");
if (error_code != ErrorCode.e_ErrSuccess)
{
```

```
    return;
}

// Get the first page of the document.
PDFPage page = doc.GetPage(0);

// Parse page.
page.StartParse((int)foxit.pdf.PDFPage.ParseFlags.e_ParsePageNormal, null,
false);
```

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

```
int width = (int)(page.GetWidth());
int height = (int)(page.GetHeight());
Matrix2D matrix = page.GetDisplayMatrix(0, 0, width, height,
page.GetRotation());

// Prepare a bitmap for rendering.
System.Drawing.Bitmap bitmap = new System.Drawing.Bitmap(width, height,
System.Drawing.Imaging.PixelFormat.Format32bppArgb);
Graphics draw = Graphics.FromImage(bitmap);
draw.Clear(Color.White);

// Render page
Renderer render = new Renderer(bitmap, false);
render.StartRender(page, matrix, null);

// Add the bitmap to image and save the image.
foxit.common.Image image = new foxit.common.Image();
image.AddFrame(bitmap);
image.SaveAs("testpage.jpg");
```

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

    ***Note**: Please check whether the "fsdk.dll" and "fsdk_dotnet.dll" have been copied to the same folder with the "test_dotnet.exe". If not, you should put the dynamic libraries to the folder manually.*

11) Double-click the executable file "test_dotnet.exe" to run this project, and then the "testpage.jpg" will be generated in the current folder.

The final contents of "Program.cs" is as follow:

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Text;
using System.Drawing;

using foxit;
using foxit.common;
using foxit.common.fxcrt;
using foxit.pdf;

namespace test_dotnet
{
    class Program
    {
        static void Main(string[] args)
        {

            // 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=").
            string sn = " ";
            string key = " ";
            ErrorCode error_code = Library.Initialize(sn, key);
            if (error_code != ErrorCode.e_ErrSuccess)
            {
                return;
            }

            using (PDFDoc doc = new PDFDoc("../../Sample.pdf"))
            {
                error_code = doc.LoadW("");
                if (error_code != ErrorCode.e_ErrSuccess)
                {
                    Library.Release();
                    return;
                }

                using (PDFPage page = doc.GetPage(0))
                {
                    // Parse page.

page.StartParse((int)foxit.pdf.PDFPage.ParseFlags.e_ParsePageNormal, null, false);

                    int width = (int)(page.GetWidth());
                    int height = (int)(page.GetHeight());
                    Matrix2D matrix = page.GetDisplayMatrix(0, 0, width, height,
page.GetRotation());
```

```
                    // Prepare a bitmap for rendering.
                    System.Drawing.Bitmap bitmap = new System.Drawing.Bitmap(width,
height, System.Drawing.Imaging.PixelFormat.Format32bppArgb);
                    Graphics draw = Graphics.FromImage(bitmap);
                    draw.Clear(Color.White);

                    // Render page
                    Renderer render = new Renderer(bitmap, false);
                    render.StartRender(page, matrix, null);

                    // Add the bitmap to image and save the image.
                    foxit.common.Image image = new foxit.common.Image();
                    image.AddFrame(bitmap);
                    image.SaveAs("testpage.jpg");
                }
            }
            Library.Release();
        }
    }
}
```

REFERENCES

[1] PDF reference 1.7
http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=51502

[2] Foxit PDF SDK API reference
sdk_folder/doc/Foxit PDF SDK Dotnet 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