Foxit

DEVELOPER GUIDE

Foxit® PDF SDK

for Universal Windows Platform on Windows 10

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.

1.1 Why Foxit is your choice

Foxit is an Amazon-invested leading software provider of solutions for reading, editing, creating, organizing, and securing PDF documents. Customers choose Foxit products for the following reasons:

- **High performance** – Very fast on PDF parsing, rendering and conversion.
- **Lightweight footprint** – Do not exhaust system resource and deploys quickly.
- **Cross-platform support** – Support Microsoft Windows, Linux etc.
- **Compatibility** – ISO 32000-1/PDF 1.7 standards compliant and compatible with other PDF products.
- **Great value/affordability** – Right features at right price with email and phone support.
- **Security** - Safeguards confidential information.

In addition, Foxit products are fully supported by our dedicated support engineers if support and maintenance are purchased. Updates are released on a regular basis. Developers may focus more on their solution building rather than spending time on PDF specification. Foxit will be the right choice if you need solutions with excellent features and low cost!

1.2 Features

Foxit PDF SDK for Universal Windows Platform (UWP) on Windows 10 is a Software Development Kit which ships with simple-to-use APIs that can be accessed from Visual C++, Visual C#, Visual Basic, JavaScript, and other languages that support the Windows Runtime. It allows developers to seamlessly integrate powerful PDF technology with the apps that target the Windows 10 Universal Apps. Foxit provides the SDK libraries in the form of a Windows Runtime Component which must be installed and added to reference before using it in your UWP apps.

Foxit PDF SDK for UWP Apps on Windows 10 supports all Windows 10 devices-PC, tablet, phone and more, and it has several main features to help application developers focus on functions that they really need and reduce the development cost.

Features	
PDF Document	Open and close PDF files

PDF Page	Get some properties, and render page
PDF Text Page	Text processing in a PDF document
Bookmark	Directly locate and link to point of interest within a document
Attachment	Get attachments, and access properties of attachments.

1.2.1 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.

1.2.2 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.

1.3 About this guide

This guide is intended for the developers who need to integrate Foxit PDF SDK for Universal Windows Platform (UWP) on Windows 10 into their own applications. It aims at introducing installation package structure on Universal Windows Platform for Windows 10, basic knowledge on PDF and the usage of SDK.

2 INTRODUCTION TO PDF

2.1 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, fonts, 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.

2.2 PDF Document Structure

A PDF document is composed of one or more pages. Each page has its own specification 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 regarded as a hierarchy of objects contained in the body section of a PDF file. Displaying a PDF document in an application involves loading PDF document, parsing PDF objects, retrieving/decoding the page content and displaying/printing it on a device. Editing a PDF document requires parsing the document structure, making changes and reorganizing the PDF objects in a document. These operations could be done by a conforming PDF reader/editor or in your own applications through APIs provided by Foxit.

2.3 PDF Document Features

PDF supports a lot of features to enhance the document capability, such as document encryption, digital signatures, java script actions, form filling, layered content, multimedia support and etc. These features provide users with more flexibility in displaying, exchanging and editing documents. Currently, Foxit PDF SDK for Universal Windows Platform (UWP) on Windows 10 supports the most common features, such as PDF viewing, bookmark navigating, text selection and search, and etc. More features will be provided in the following release. Users can use Foxit PDF SDK to fulfill these advanced features in your applications.

3 GETTING STARTED

It is very easy to setup Foxit PDF SDK and see it in action! It takes just a few minutes and we will show you how to integrate Foxit PDF SDK with Universal Windows Platform (UWP) apps on Windows 10. The following sections introduce system requirements, the structure of installation package, how to apply a license, and how to run a demo.

3.1 System Requirements

Windows 10 Universal Apps:

Windows 10

Visual Studio 2015 (with Universal Windows App Development Tools) installed

The release package includes arm, x64 and x86 dynamic link libraries for Windows 10 Universal app.

3.2 What is in the Package

Download Foxit PDF SDK zip for UWP on Windows 10 and extract it to a new directory like "foxitpdfsdk_5_0_win10_uwp". The structure of the release package is shown in Figure 3-1. One thing to note is that the highlighted rectangle in the figure is the version of the SDK. Here the SDK version is 5.0, so it shows 5_0. Other highlighted rectangles have the same meaning in this guide. This package contains the following folders:

docs: API references, Developer Guide

lib: Windows runtime component and license files

samples: sample projects and demos

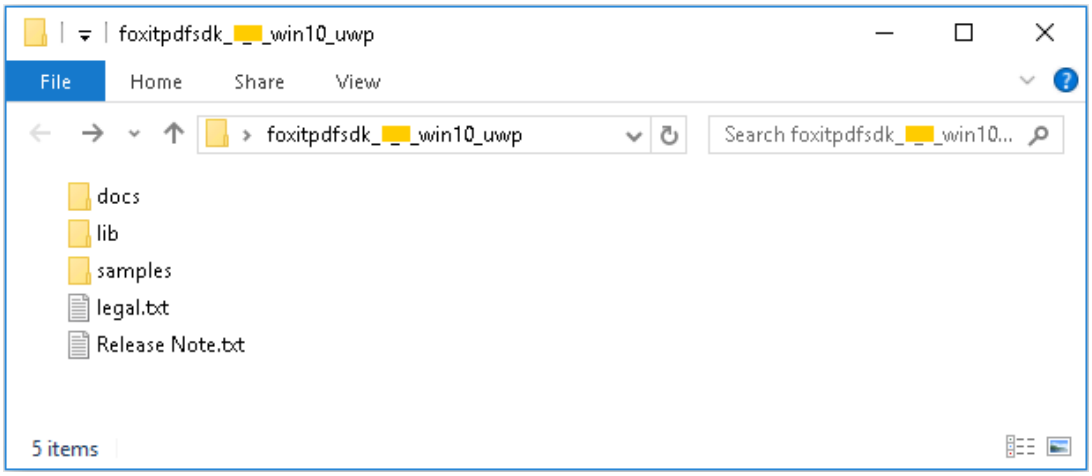

Figure 3-1

Foxit provides the SDK libraries in the form of a Windows runtime component (**FoxitPDFSDK.vsix**) under "lib" folder which is an extension file for Visual Studio. Before using it in your UWP apps, you should install it first, and then add it to reference.

3.3 How to apply a license

It is necessary for applications to initialize and unlock Foxit PDF SDK license before calling any APIs. The function Foxit.Library.Load(licenseKey, unlockCode) is provided to unlock Foxit PDF SDK license. An example of applying a license is shown below. The parameter "licenseKey" can be found in the "gsdk_sn.txt" (the string after "SN=") and the "unlockCode" can be found in the "gsdk_key.txt" (the string after "Sign=").

```
// Initialize and unlock Foxit PDF SDK license.
Foxit.ErrorCode ret;
Foxit.Library.Load(licenseKey, unlockCode);
ret = Foxit.Library.GetLastError();
Foxit.Library.LoadSystemFonts();
ret = Foxit.Library.GetLastError();
```

3.4 How to run a demo

Foxit PDF SDK for Universal Windows Platform (UWP) on Windows 10 provides a simple viewer demo in "samples" folder.

demo_view

demo_view project provides an example for developers on how to implement a simple PDF viewer with C# on Universal Windows Platform for Windows 10 using Foxit PDF SDK APIs. To run the demo in Visual Studio 2015, follow the steps below: (in this guide, we use an x86 simulator as an example to run the project)

a) Install Foxit PDF SDK extension. Double-click **FoxitPDFSDK.vsix** under "lib" folder, choose **Yes** in the "*User Account Control*" dialog as shown in Figure 3-2, and then click on **Install** in the "*VSIX Installer*" dialog to allow Foxit PDF SDK extension to be installed. The screenshot is shown in Figure 3-3. Note: If Visual Studio 2015 has been opened, please restart it after installing Foxit PDF SDK extension.

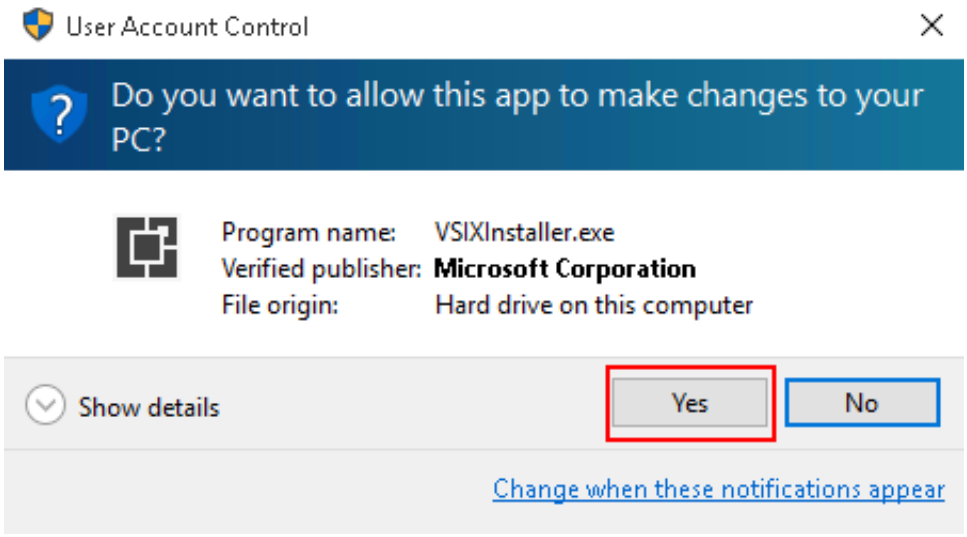

Figure 3-2

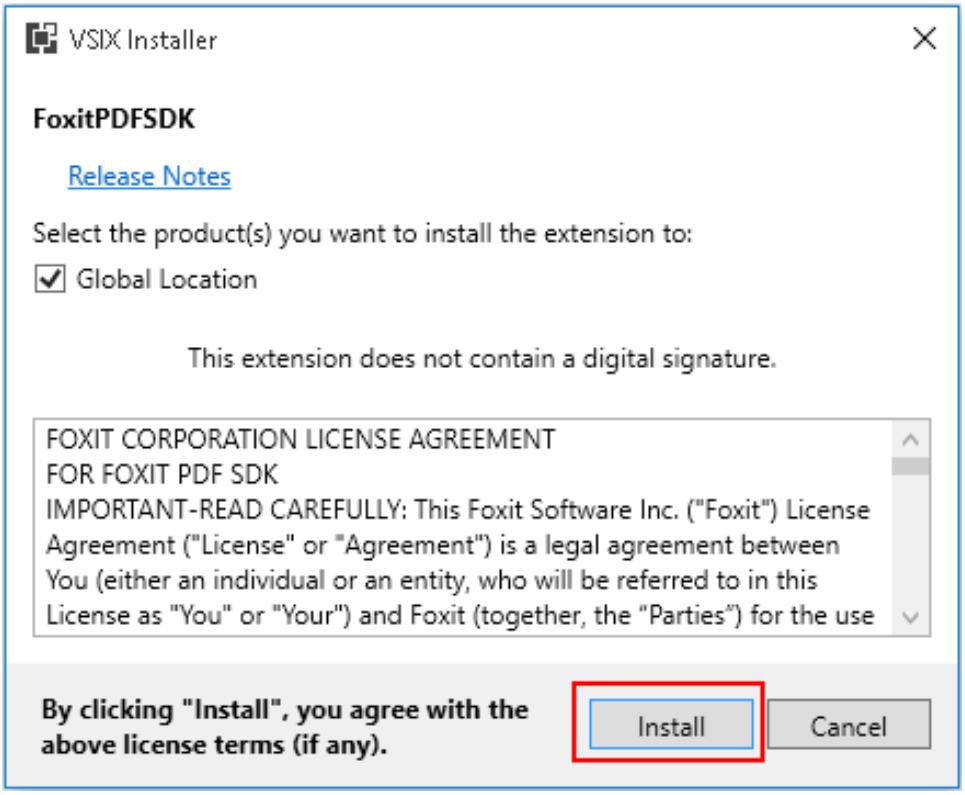

Figure 3-3

Note: *If you already have installed a version of Foxit PDF SDK extension for Windows Universal apps, you might get an error message stating "**This extension is already installed to all applicable products.**" In this case, please uninstall the old version in Visual Studio 2015 by*

*following the steps: Open menu "**Tools** -> **Extensions and Updates…**" Under the menu **Installed** -> **All**, you should be able to find "Foxit PDF SDK" extension. Select it and click **Uninstall**.*

b) Open the **demo_view.sln** file under "samples/demo_view" folder in Visual Studio 2015. And then check whether the Foxit PDF SDK extension has already been installed successfully in Visual Studio. Click on **Tools** -> **Extensions and Updates…** in the menu of Visual Studio, you will see the installed extension as shown in Figure 3-4.

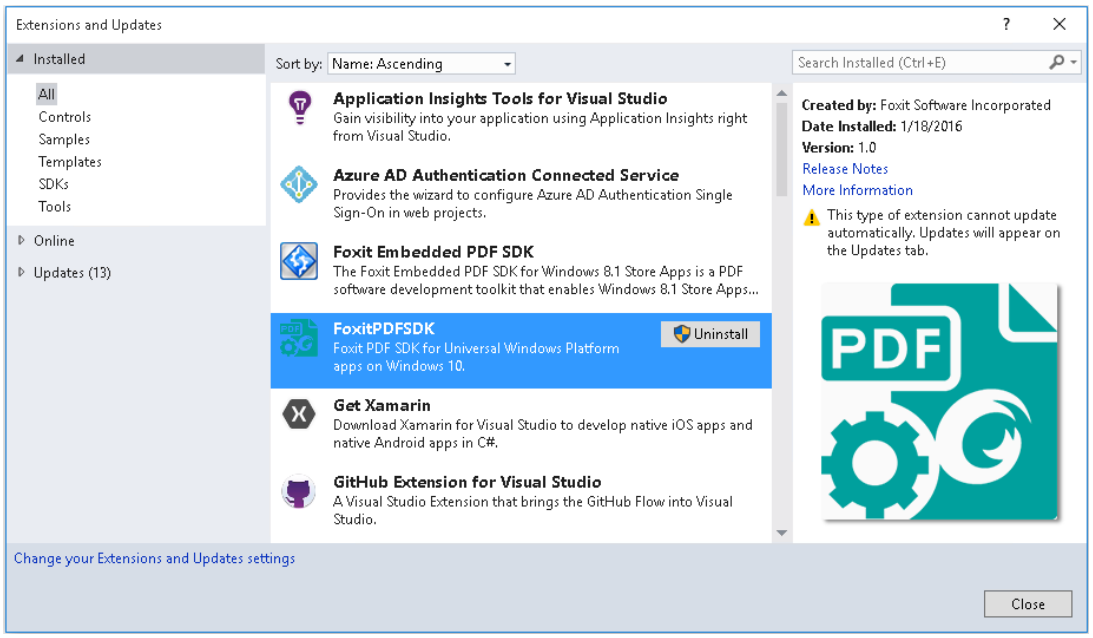

Figure 3-4

c) Change the build architecture of the project. Click on **Build** -> **Configuration Manager** and choose **x86** for the "Active solution platform" as shown in Figure 3-5.

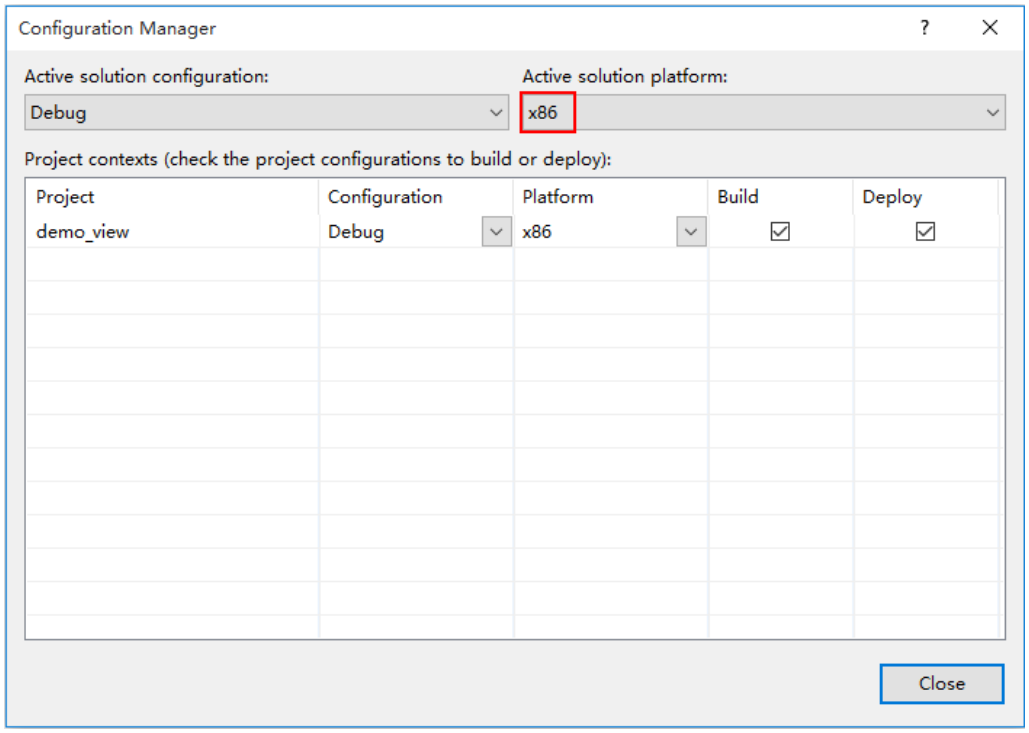

Figure 3-5

Note: *There are three active solution platforms: x86, x64, and arm. You should choose the proper platform for the build architecture according to the system you used to run the project. For example, if you will run the demo in an arm device, please choose ARM for the "Active solution platform".*

d) Click on **Simulator** as shown in Figure 3-6 to build and run the demo. The screenshot of the demo is shown in Figure 3-7.

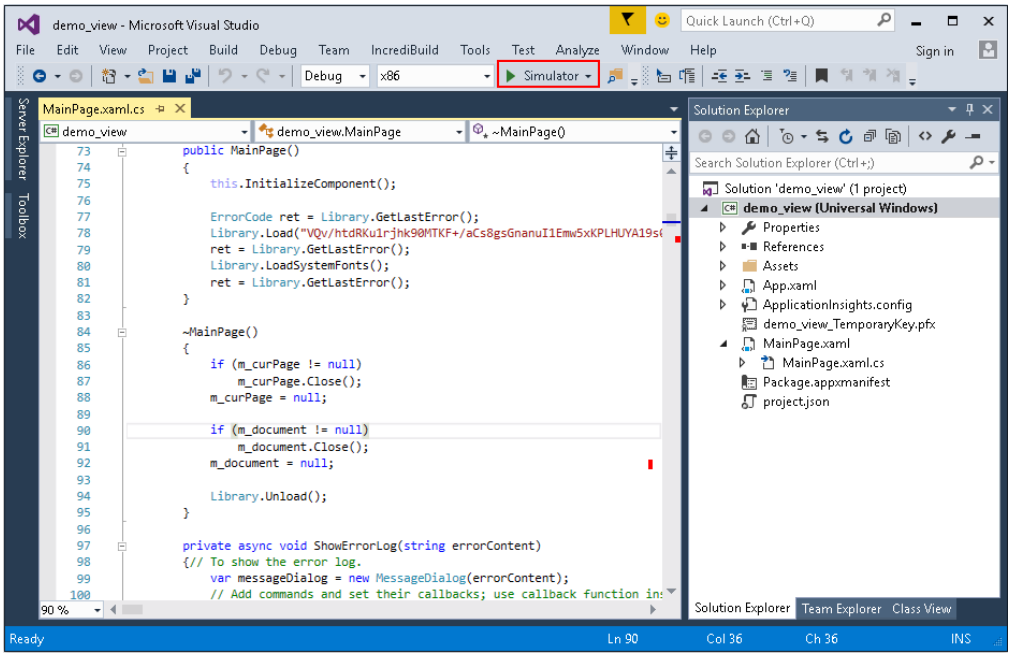

Figure 3-6

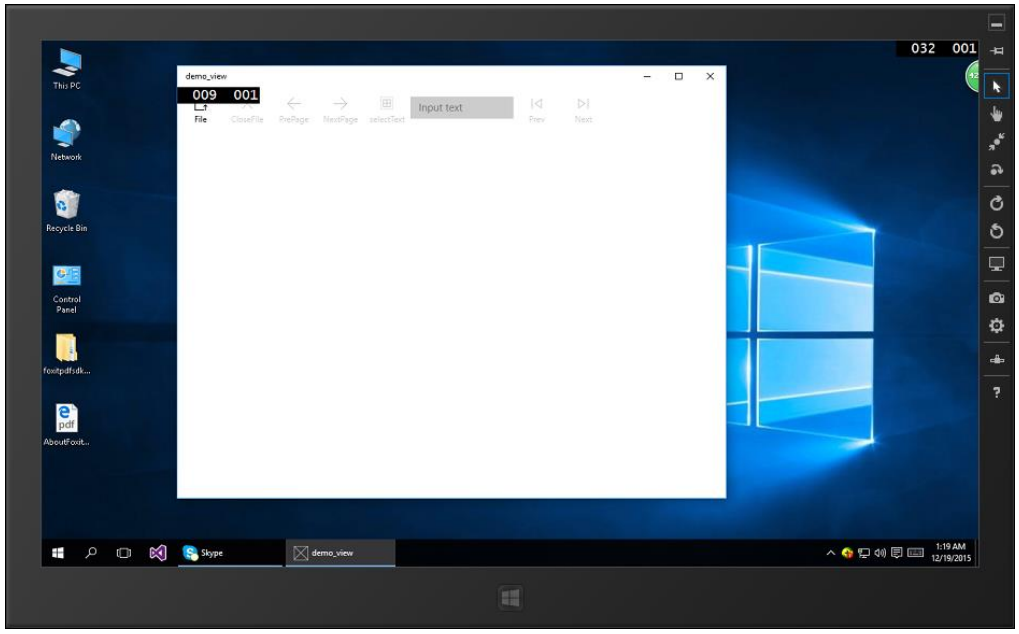

Figure 3-7

e) Click on **File** to open a PDF file. Here, we open a PDF document named “AboutFoxit.pdf”. The “AboutFoxit.pdf” will be displayed as shown in Figure 3-8.

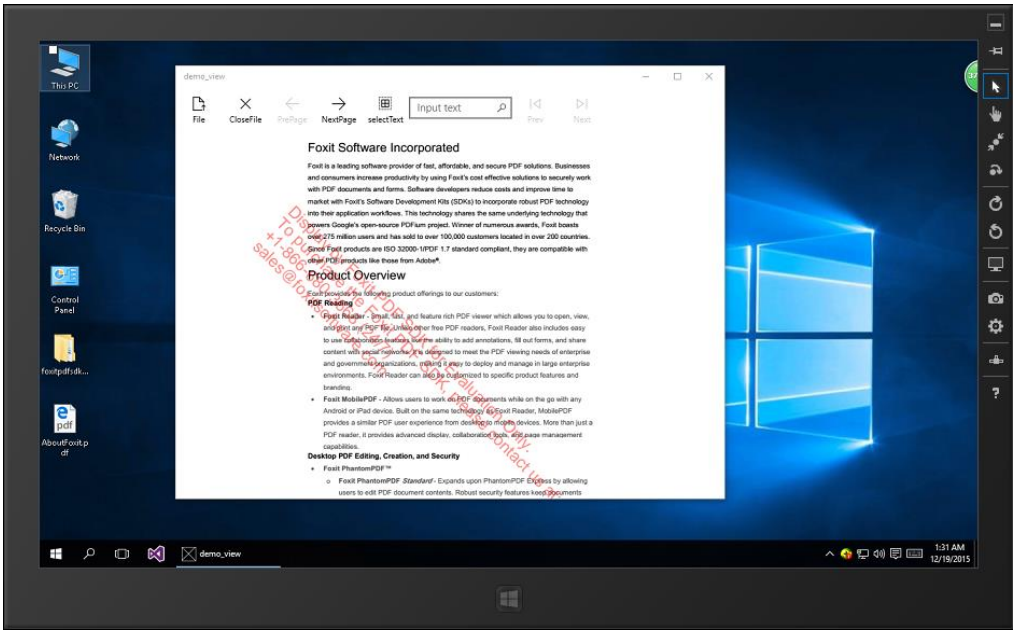

Figure 3-8

f) This demo provides the features like rendering a PDF document, page turning, text selection and search. For example, click the *search box*, type word “Foxit”, and then press the **Enter** key, the first search result will be highlighted as shown in Figure 3-9. Click the **Next** button, you can find the next one you search for.

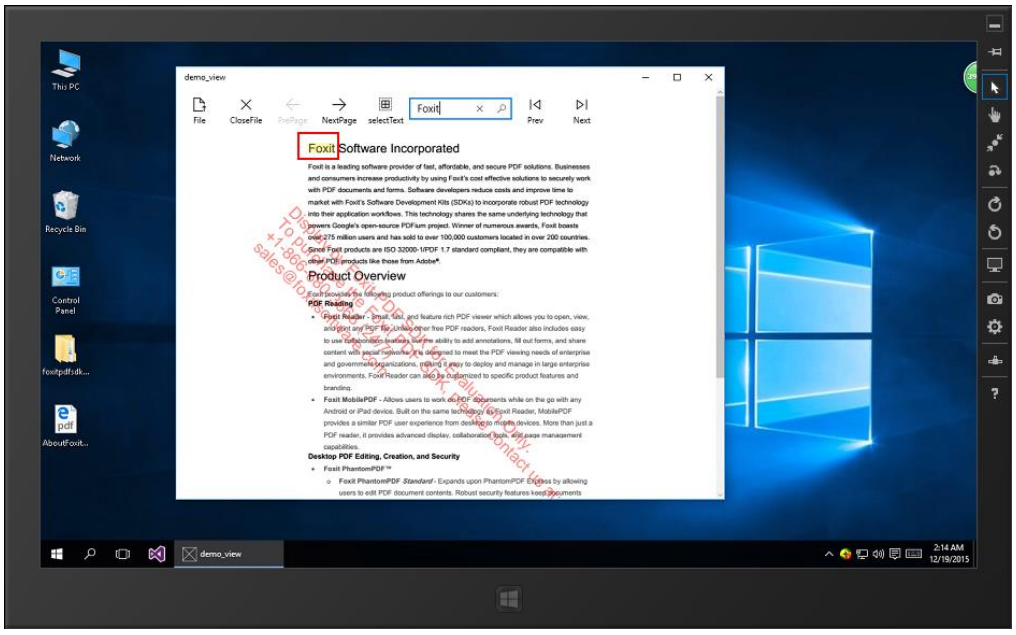

Figure 3-9

4 CREATE YOUR OWN PROJECT

In this section, we will show you how to create your own UWP project on Windows 10 platform and how to render a PDF document using Foxit PDF SDK APIs. Create a new Windows Universal project in Visual Studio 2015 called “test_uwp”, and we will be using C# and a blank app, it is shown in Figure 4-1.

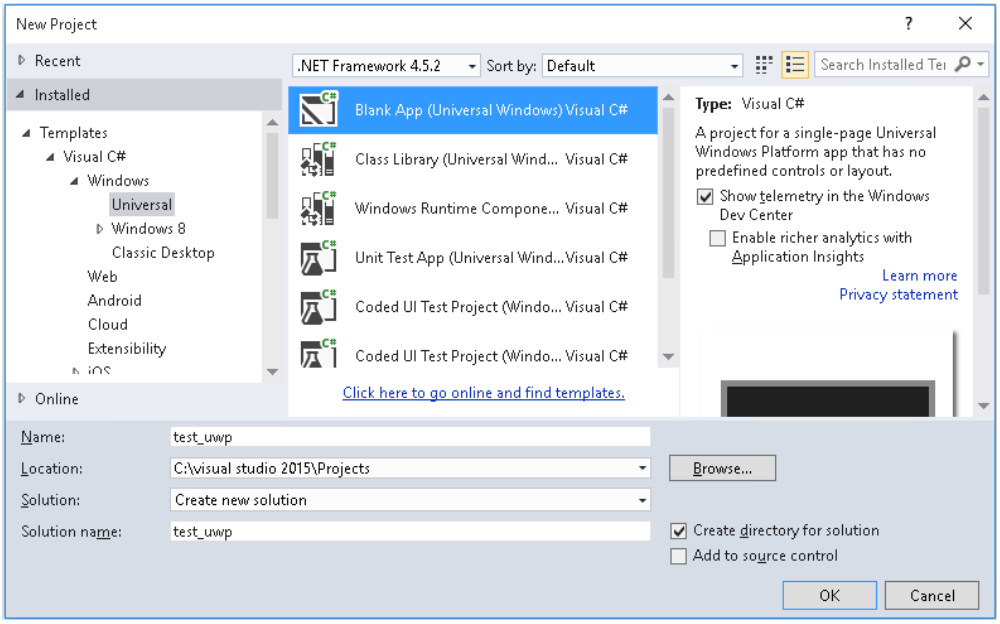

Figure 4-1

To run this project in Visual Studio 2015, please follow the steps below: (In this guide, we also use an x86 simulator as an example to run the project)

a) Add the extension “FoxitPDFSDK” to **References**. In order to use a component in the project, you must first add a reference to it.

   i. In **Solution Explorer**, right-click the project node and click **Add Reference** as shown in Figure 4-2.

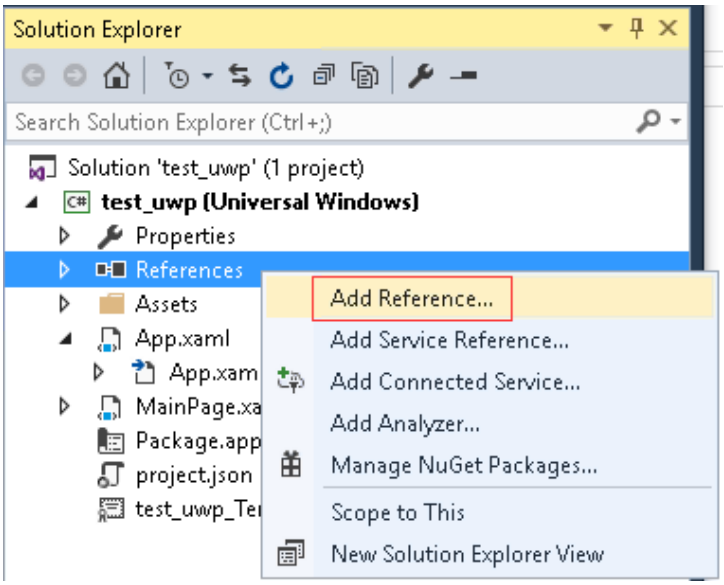

Figure 4-2

ii. In the **Reference Manager** dialog, select the **Universal Windows** -> **Extensions** tab, check the box next to "FoxitPDFSDK", and then click **OK**. It is shown in Figure 4-3.

Note: *Here, we assume that you have already installed Foxit PDF SDK extension. If not, please install it first referring to "How to run a demo".*

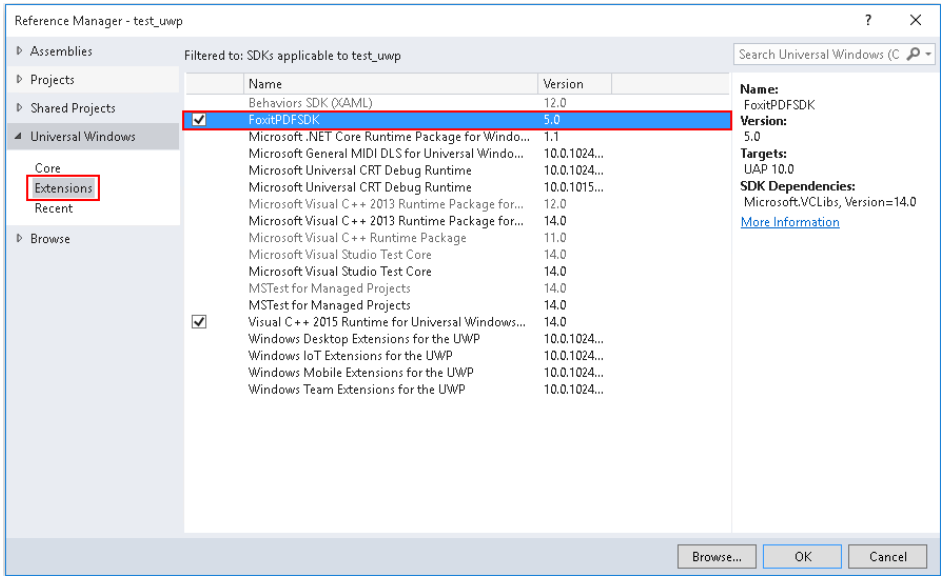

Figure 4-3

b) Change the build architecture of the project. Click on **Build** -> **Configuration Manager** and select **x86** for the "Active solution platform" as shown in Figure 4-4.

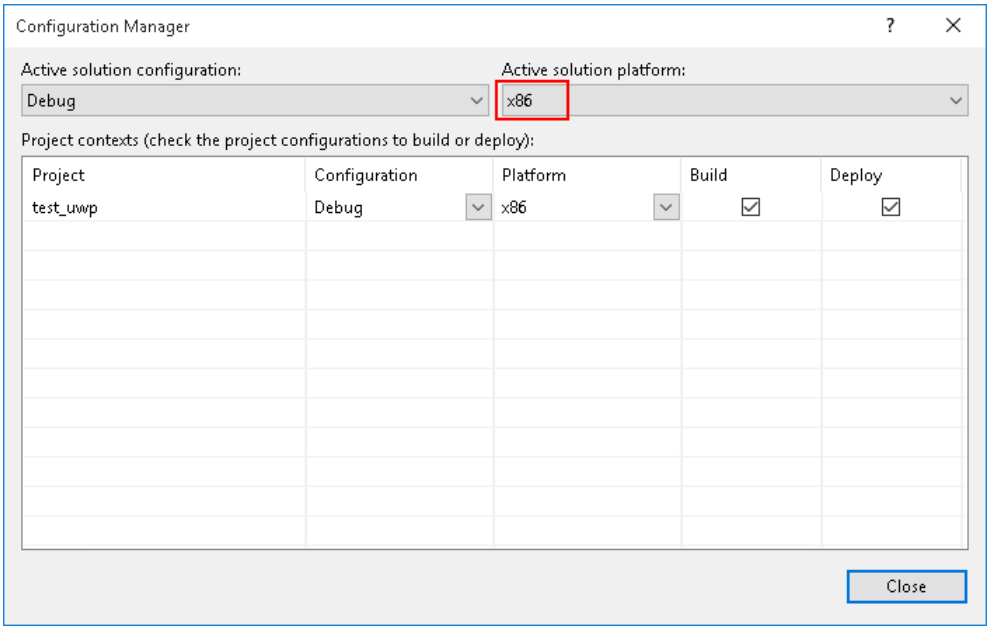

Figure 4-4

Note: *There are three active solution platforms: x86, x64, and arm. You should choose the proper platform for the build architecture according to the system you used to run the project. For example, if you will run the demo in an arm device, please choose ARM for the "Active solution platform".*

c) Construct the code to build a PDF application which uses Foxit PDF SDK APIs to display a PDF document.

1) Open up "MainPage.xaml" and find the **Grid** element. Add some row definitions as follows:

```
<Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
       <Button Content="Open File" Click="Button_Click"
       HorizontalAlignment="Center" FontSize="30" Width="150" Height="80">
       </Button>
        <Grid x:Name="imagePanel">
            <Image x:Name="image" Stretch="Fill"></Image>
        </Grid>
</Grid>
```

Here, we add a button with a click event "Button_Click" to open a document, and an image panel to display the PDF document.

2) Initialize Foxit PDF SDK libraries. We do this in the constructor of the MainPage in "MainPage.xaml.cs". Open up "MainPage.xaml.cs", and add the codes as follows:

```
public MainPage()
        {
            this.InitializeComponent();

            string license_id;
            string unlockCode;

            Foxit.ErrorCode ret;
            Foxit.Library.Load(license_id, unlockCode);
            ret = Foxit.Library.GetLastError();
            Foxit.Library.LoadSystemFonts();
            ret = Foxit.Library.GetLastError();
        }
```

The value of the “license_id” can be found in the “gsdk_sn.txt” (the string after “SN=”), and the value of the “unlockCode” can be found in the “gsdk_key.txt” (the string after “Sign=”).

3) Include the following namespaces that we need for the file picker, storage stream, storage file and writing bitmap.

```
using Windows.Storage;
using Windows.Storage.Pickers;
using Windows.Storage.Streams;
using Windows.UI.Xaml.Media.Imaging;
```

4) Add “async” to the “Button_Click”. The “Button_Click” needs to be async in order to handle some of the asynchronous API’s needed. And then choose a PDF document from the file picker. Add the following codes to the “Button_Click” function.

```
private async void Button_Click(object sender, RoutedEventArgs e)
       {
              FileOpenPicker openPicker = new FileOpenPicker();
              openPicker.ViewMode = PickerViewMode.List;
              openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
              openPicker.FileTypeFilter.Add(".pdf");

              StorageFile m_pdfFile = await openPicker.PickSingleFileAsync();
              if (m_pdfFile == null)
              return;
       }
```

5) Load the selected PDF document and get the current PDF page.

```
Foxit.PDF.Document m_document = new Foxit.PDF.Document();
bool bRet = await m_document.LoadAsync(m_pdfFile, "", 0);
if (bRet)
{
  int pagecount = m_document.CountPages();
  if (pagecount < 1 || Foxit.Library.GetLastError() != Foxit.ErrorCode.Success)
  {
      return;
  }
}

Foxit.Pause pause = null;
Foxit.PDF.Page m_curPage = await m_document.LoadPageAsync(0, 0, pause);
if (m_curPage.pointer == 0)
  return;
```

6) Render and display the current PDF page.

```
// Get the page size of the current PDF page.
float m_PageWidth = m_curPage.GetWidth();
float m_PageHeight = m_curPage.GetHeight();

Foxit.PixelSource bitmap = new Foxit.PixelSource();
bitmap.Width = (int)m_curPage.GetWidth();
bitmap.Height = (int)m_curPage.GetHeight();
Foxit.Matrix matrix = m_curPage.GetDisplayMatrix(0, 0, (int)m_PageWidth,
(int)m_PageHeight, 0);
IRandomAccessStream randomStream = await m_curPage.RenderPageAsync(bitmap,
matrix, (uint)Foxit.PDF.RenderFlags.Annot, null);
if (null != randomStream)
{
     WriteableBitmap bmpImage = new WriteableBitmap((int)m_PageWidth,
(int)m_PageHeight);
     bmpImage.SetSource(randomStream);
     bmpImage.Invalidate();
     image.Width = (int)m_PageWidth;
     image.Height = (int)m_PageHeight;
     image.Source = bmpImage;
}
```

d) Click on "**Simulator**" to build and run the project. The screenshot of the project is shown in Figure 4-5.

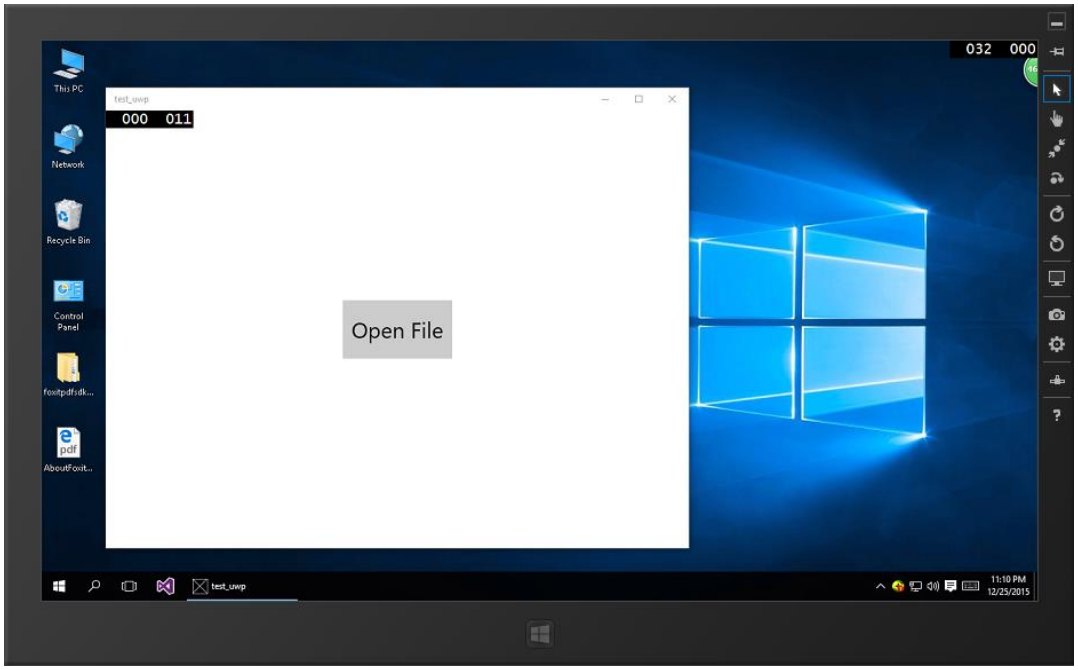

Figure 4-5

e) Click on **Open File** to choose and open a PDF document. A selecting window will be popped up as shown in Figure 4-6 when you click on **Open File.**

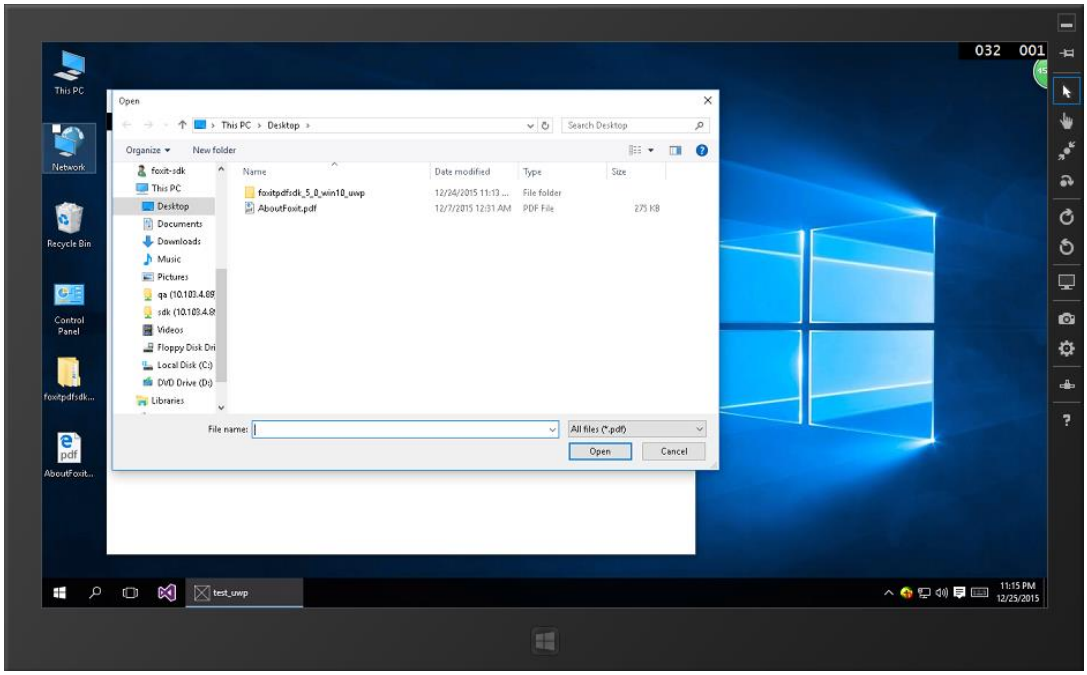

Figure 4-6

f) The selected PDF document will be displayed as shown in Figure 4-7.

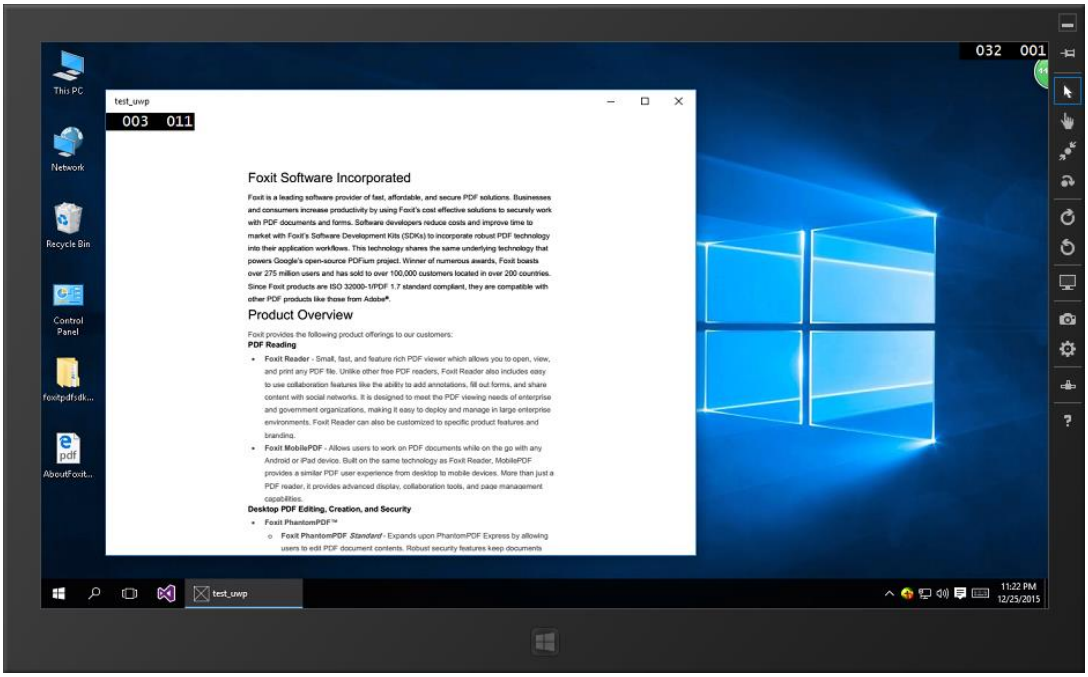

Figure 4-7

Complete program:

The following codes show MainPage.xaml and MainPage.xaml.cs in their entirety.

MainPage.xaml

```
<Page
    x:Class="test_uwp.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:test_uwp"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d">

    <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

        <Button Content="Open File" Click="Button_Click" HorizontalAlignment="Center"
        FontSize="30" Width="150" Height="80"> </Button>

        <Grid x:Name="imagePanel">
            <Image x:Name="image" Stretch="Fill"></Image>
        </Grid>

    </Grid>
</Page>
```

MainPage.xaml.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Runtime.InteropServices.WindowsRuntime;
using Windows.Foundation;
using Windows.Foundation.Collections;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Controls.Primitives;
using Windows.UI.Xaml.Data;
using Windows.UI.Xaml.Input;
using Windows.UI.Xaml.Media;
using Windows.UI.Xaml.Navigation;

// Include the naemspaces.
using Windows.Storage;
using Windows.Storage.Pickers;
using Windows.Storage.Streams;
using Windows.UI.Xaml.Media.Imaging;


// The Blank Page item template is documented at
http://go.microsoft.com/fwlink/?LinkId=402352&clcid=0x409

namespace test_uwp
{
    /// <summary>
    /// An empty page that can be used on its own or navigated to within a Frame.
    /// </summary>
    public sealed partial class MainPage : Page
    {
        public MainPage()
        {
            this.InitializeComponent();

            // Initialize Foxit PDF SDK libraries.
            // The value of "license_id" can be found in the “gsdk_sn.txt”.
            // The value of "unlockCode" can be found in the “gsdk_key.txt”.
            string license_id = "";
            string unlockCode = "";

            Foxit.ErrorCode ret;
            Foxit.Library.Load(license_id, unlockCode);
            ret = Foxit.Library.GetLastError();
            Foxit.Library.LoadSystemFonts();
            ret = Foxit.Library.GetLastError();
        }

private async void Button_Click(object sender, RoutedEventArgs e)
        {
            // Choose a PDF document from the file picker.
            FileOpenPicker openPicker = new FileOpenPicker();
            openPicker.ViewMode = PickerViewMode.List;
            openPicker.SuggestedStartLocation = PickerLocationId.Desktop;
            openPicker.FileTypeFilter.Add(".pdf");
                return;
```

```
            StorageFile m_pdfFile = await openPicker.PickSingleFileAsync();
            if (m_pdfFile == null)
                return;

            // Load the selected PDF document.
            Foxit.PDF.Document m_document = new Foxit.PDF.Document();
            bool bRet = await m_document.LoadAsync(m_pdfFile, "", 0);
            if (bRet)
            {
                int pagecount = m_document.CountPages();
                if (pagecount < 1 || Foxit.Library.GetLastError() !=
                    Foxit.ErrorCode.Success)
                {
                      return;
                }
            }

            // Get the current PDF page.
            Foxit.Pause pause = null;
            Foxit.PDF.Page m_curPage = await m_document.LoadPageAsync(0, 0,
                  pause);
            if (m_curPage.pointer == 0)
                return;

            // Get the page size of the current PDF page.
            float m_PageWidth = m_curPage.GetWidth();
            float m_PageHeight = m_curPage.GetHeight();

            // Render and display the current PDF page.
            Foxit.PixelSource bitmap = new Foxit.PixelSource();
            bitmap.Width = (int)m_curPage.GetWidth();
            bitmap.Height = (int)m_curPage.GetHeight();
            Foxit.Matrix matrix = m_curPage.GetDisplayMatrix(0, 0,
              (int)m_PageWidth, (int)m_PageHeight, 0);
            IRandomAccessStream randomStream = await
              m_curPage.RenderPageAsync(bitmap, matrix,
              (uint)Foxit.PDF.RenderFlags.Annot, null);
            if (null != randomStream)
            {
                WriteableBitmap bmpImage = new
                 WriteableBitmap((int)m_PageWidth, (int)m_PageHeight);
                bmpImage.SetSource(randomStream);
                bmpImage.Invalidate();
                image.Width = (int)m_PageWidth;
                image.Height = (int)m_PageHeight;
                image.Source = bmpImage;
            }
        }
    }
}
```

5 FAQ

1. **What is the price of Foxit PDF SDK for Universal Windows Platform (UWP) on Windows 10?**
   To receive a price quotation, please send a request to sales@foxitsoftware.com or call Foxit sales at 1-866-680-3668.

2. **How can I activate it after purchasing Foxit PDF SDK for Universal Windows Platform (UWP) on Windows 10?**
   There are detailed descriptions on how to apply a license in the section 3.3. You can refer to the descriptions to activate a license.

3. **How can I look for technical support when I try Foxit PDF SDK for Universal Windows Platform (UWP) on Windows 10?**
   You can send email to support@foxitsoftware.com for any questions or comments or call our support at 1-866-693-6948.

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/docs/Foxit PDF SDK (Win10UWP) API Reference.chm

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

GLOSSARY OF TERMS & ACRONYMS

catalog	The primary dictionary object containing references directly or indirectly to all other objects in the document, with the exception that there may be objects in the trailer that are not referred to by the catalog
character	Numeric code representing an abstract symbol according to some defined character encoding rule
developer	Any entity, including individuals, companies, non-profits, standards bodies, open source groups, etc., who are developing standards or software to use and extend ISO 32000-1
dictionary object	An associative table containing pairs of objects, the first object being a name object serving as the key and the second object serving as the value and may be any kind of object including another dictionary
direct object	Any object that has not been made into an indirect object
FDF file	File conforming to the Forms Data Format containing form data or annotations that may be imported into a PDF file
filter	An optional part of the specification of a stream object, indicating how the data in the stream should be decoded before it is used
font	Identified collection of graphics that may be glyphs or other graphic elements
function	A special type of object that represents parameterized classes, including mathematical formulas and sampled representations with arbitrary resolution
glyph	Recognizable abstract graphic symbol that is independent of any specific design
indirect object	An object that is labelled with a positive integer object number followed by a non-negative integer generation number followed by object and having end object after it
integer object	Mathematical integers with an implementation specified interval centred at 0 and written as one or more decimal digits optionally preceded by a sign

name object	An atomic symbol uniquely defined by a sequence of characters introduced by a SOLIDUS (/), (2Fh) but the SOLIDUS is not considered to be part of the name
null object	A single object of type null, denoted by the keyword null, and having a type and value that are unequal to those of any other object
numeric object	An integer object representing mathematical integers or a real object representing mathematic real numbers
object	Basic data structure from which PDF files are constructed. Types of objects in PDF include: boolean, numerical, string, name, array, dictionary, stream and null
object reference	An object value used to allow one object to refer to another; that has the form “<n> <m> R” where <n> is an indirect object number, <m> is its version number and R is the uppercase letter R
PDF	Portable Document Format file format defined by this specification [ISO 32000-1]
real object	This object used to approximate mathematical real numbers, but with limited range and precision and written as one or more decimal digits with an optional sign and a leading, trailing, or embedded PERIOD (2Eh) (decimal point)
rectangle	A specific array object used to describe locations on a page and bounding boxes for a variety of objects and written as an array of four numbers giving the coordinates of a pair of diagonally opposite corners, typically in the form [llx lly urx ury] specifying the lower-left x, lower-left y, upper-right x, and upper-right y coordinates of the rectangle, in that order
stream object	This object consists of a dictionary followed by zero or more bytes bracketed between the keywords stream and endstream
string object	This object consists of a series of bytes (unsigned integer values in the range 0 to 255). String objects are not integer objects, but are stored in a more compact format