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.
Annotation	Create, edit and remove annotations.
Signature	Sign a PDF document, verify a signature, add or delete a signature field.

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 selecting/copying/searching, annotations, signature, 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 run a demo, and how to create your own project.

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_3_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 5.3, so it shows 5_3. 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

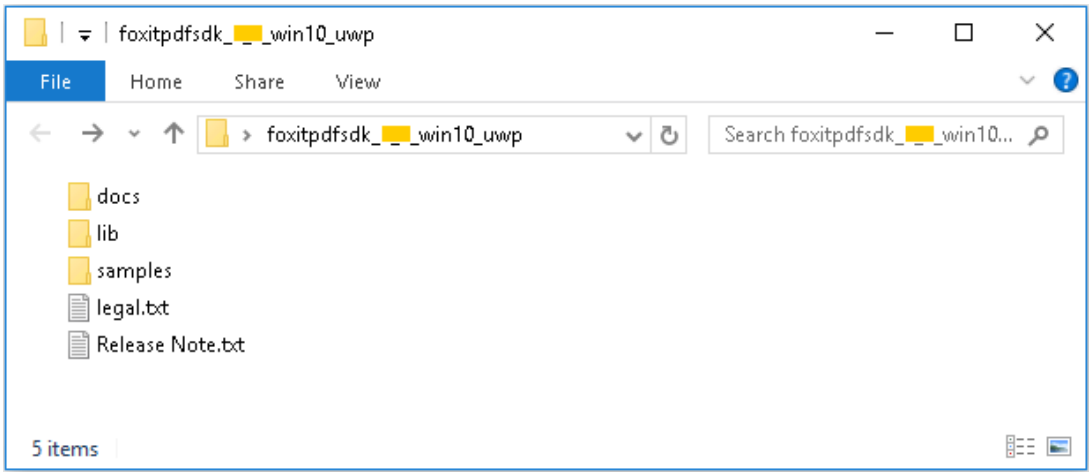

Figure 3-1

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 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 the local machine with x86 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.

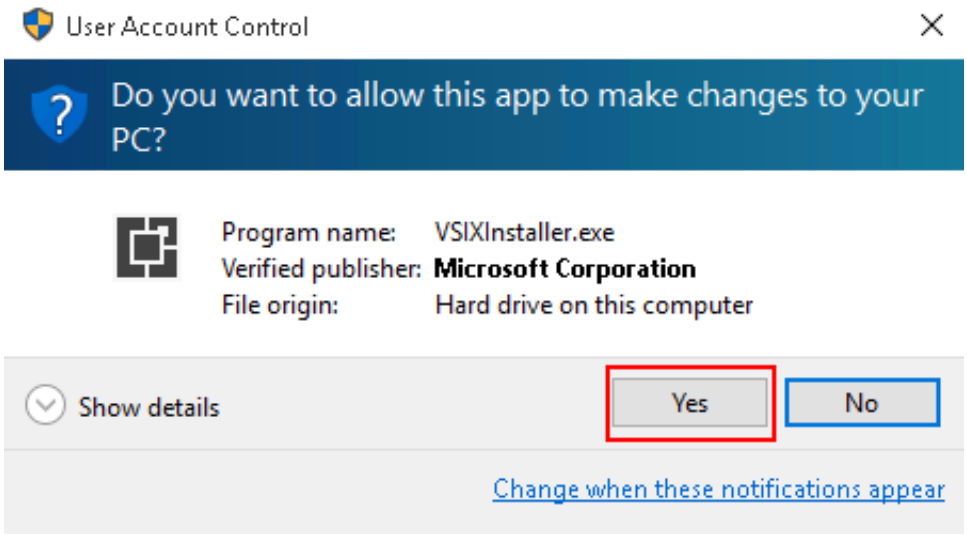

Figure 3-2

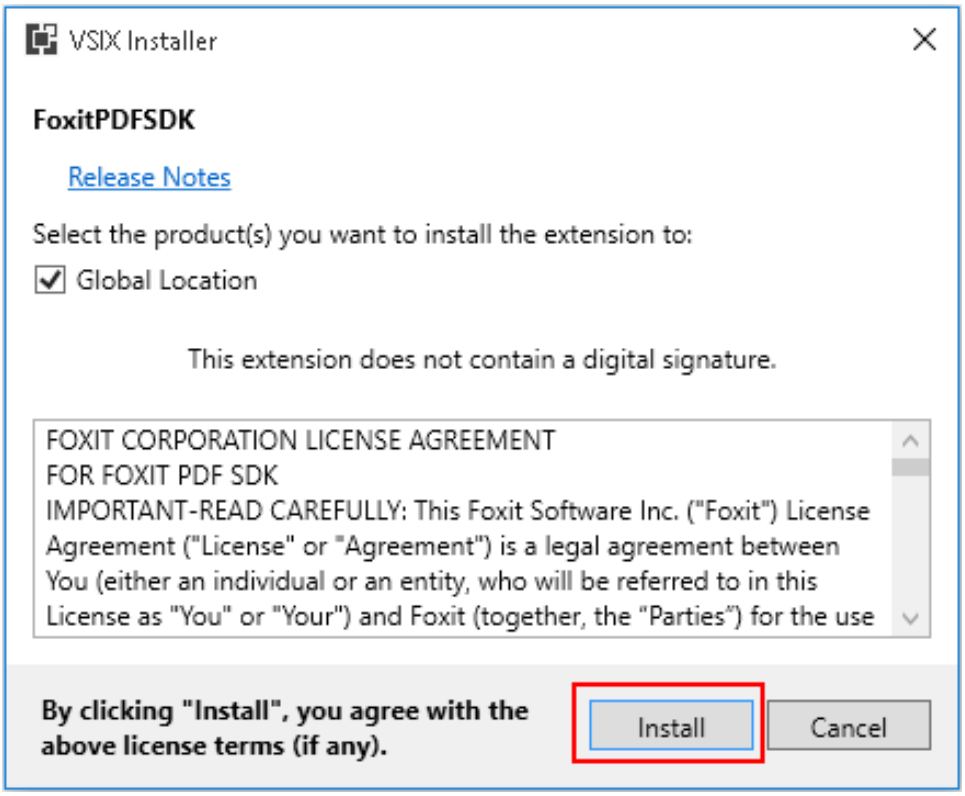

Figure 3-3

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.

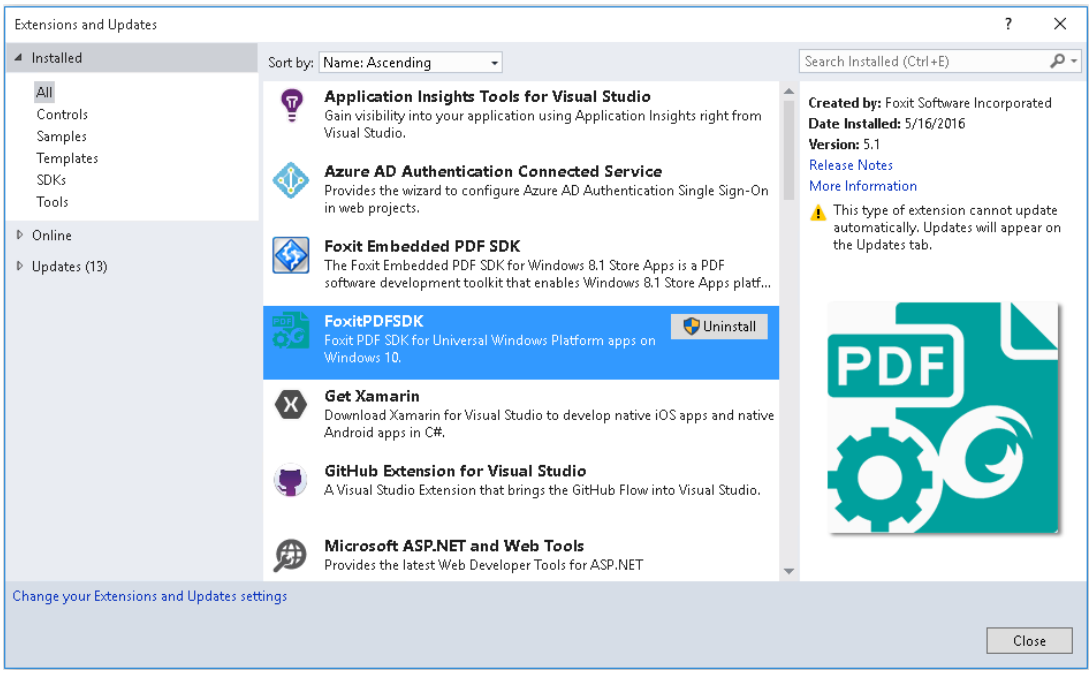

Figure 3-4

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.

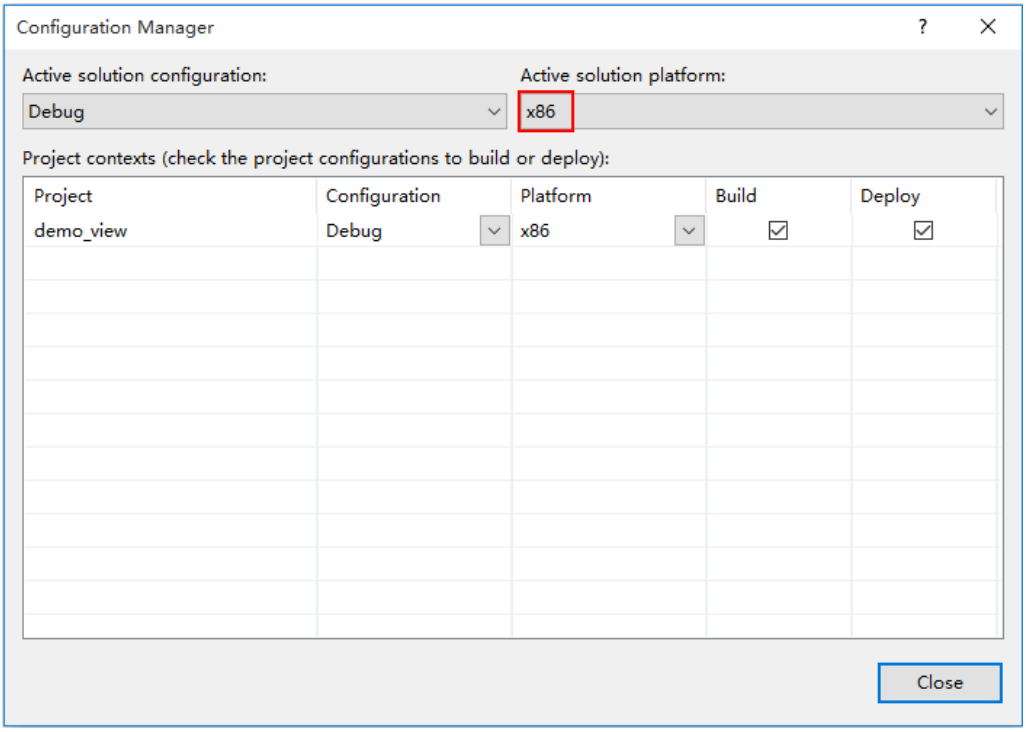

Figure 3-5

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 **Local Machine** as shown in Figure 3-6 to build and run the demo. The screenshot of the demo is shown in Figure 3-7.

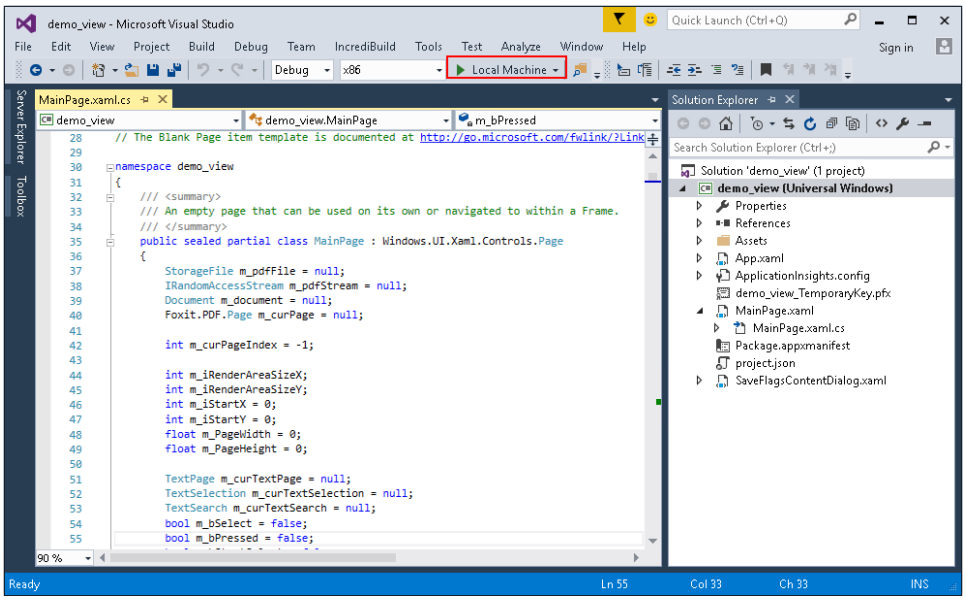

Figure 3-6

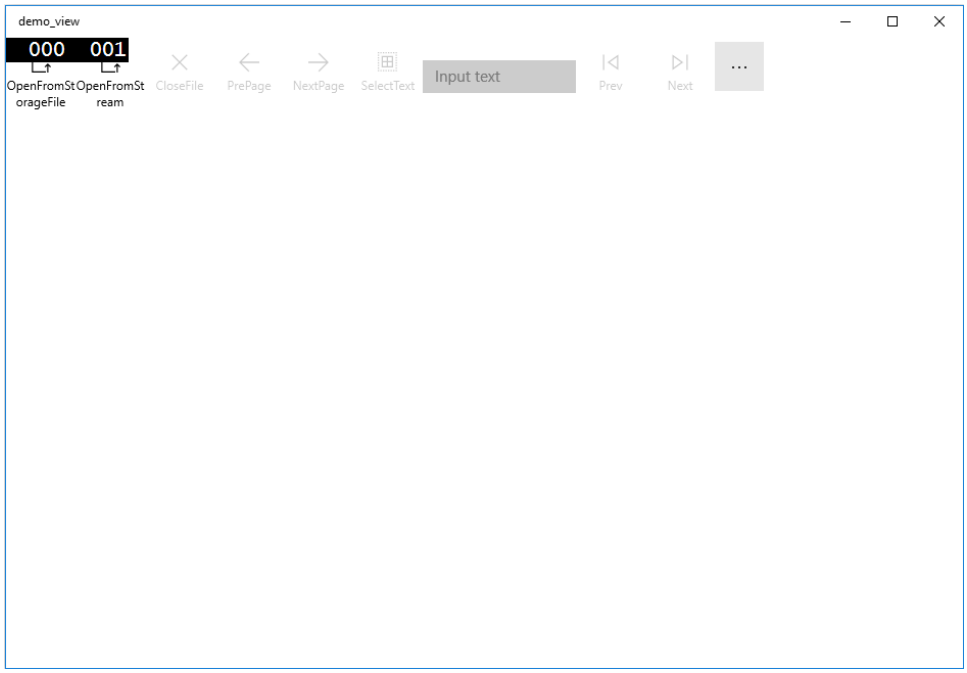

Figure 3-7

e) Click on **OpenFromStorageFile** 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.

Note: *This demo offers two ways to open a PDF file: one is from storage file, and the other is from stream.*

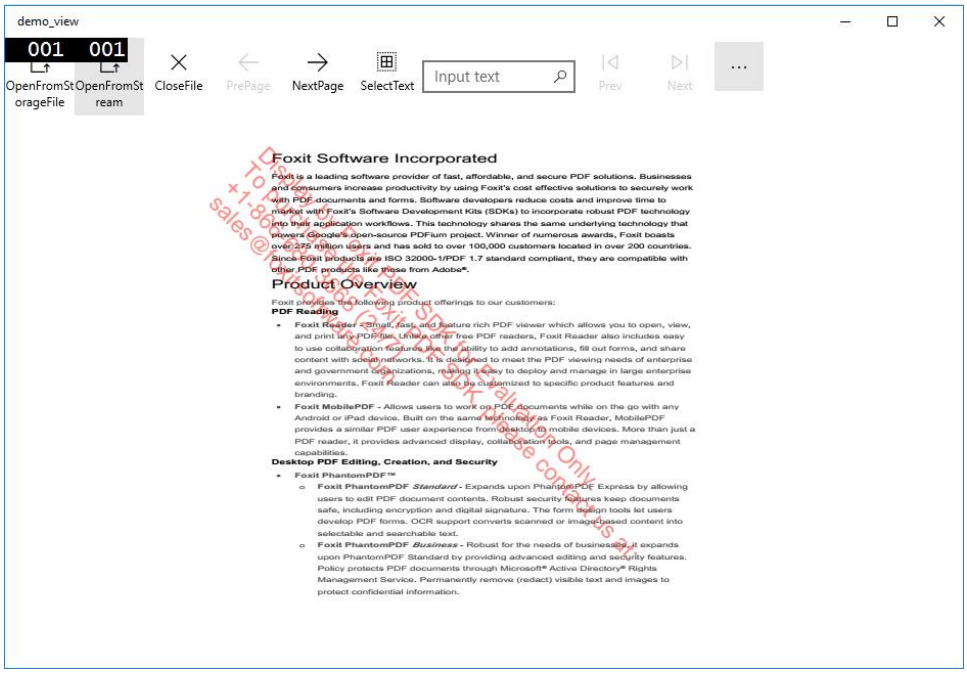

Figure 3-8

f) This demo provides the features like rendering a PDF document, page turning, text selection and search. Click the "**…**" to see the "save as" feature as shown in Figure 3-9.

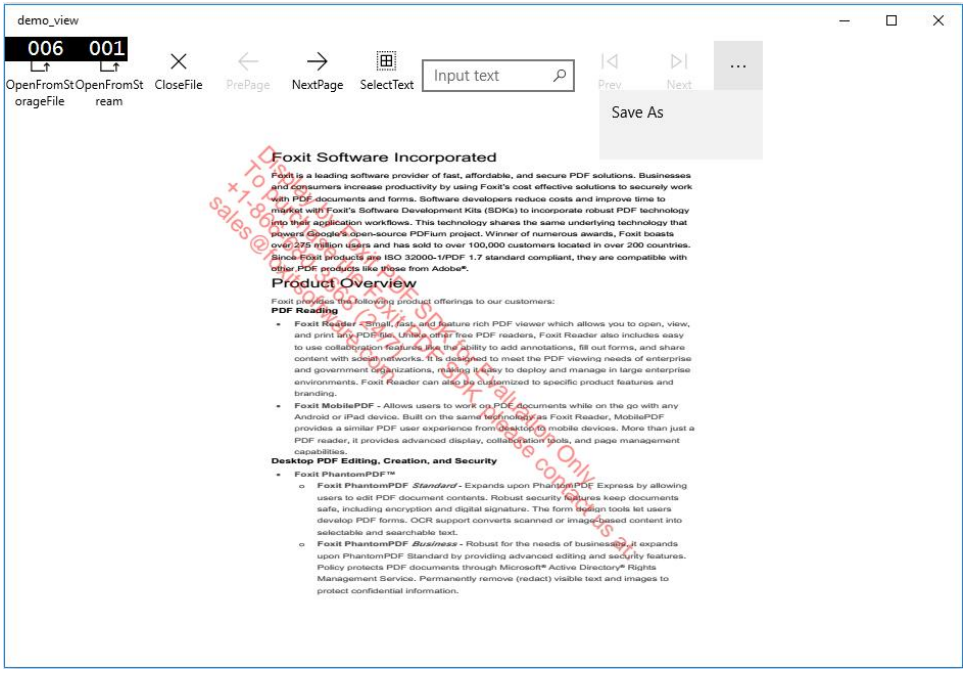

Figure 3-9

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 3-10. Click the **Next** button, you can find the next one you search for.

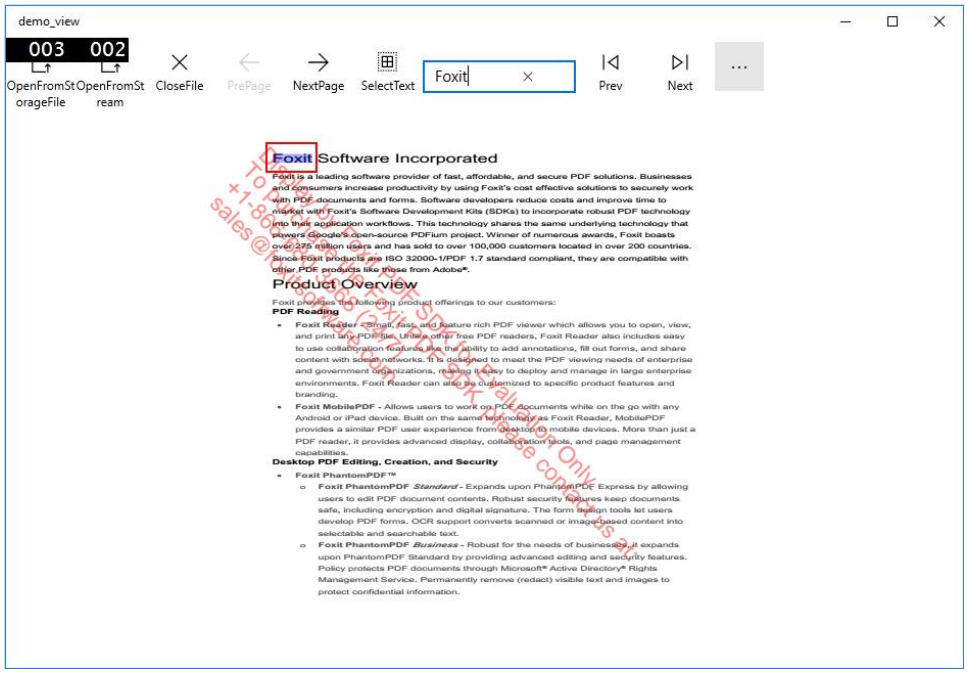

Figure 3-10

Click the "**…**", and then click **Save As**, the "SaveFlags" window will be popped up as shown in Figure 3-11. You can select the flags to save the PDF file.

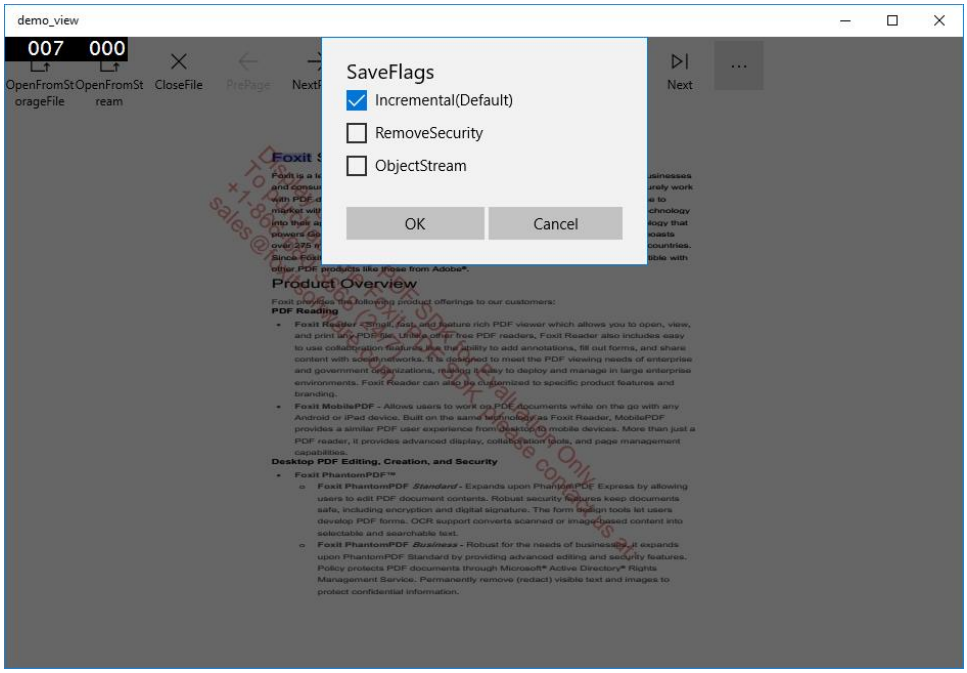

Figure 3-11

3.4 How to 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, which is shown in Figure 3-12.

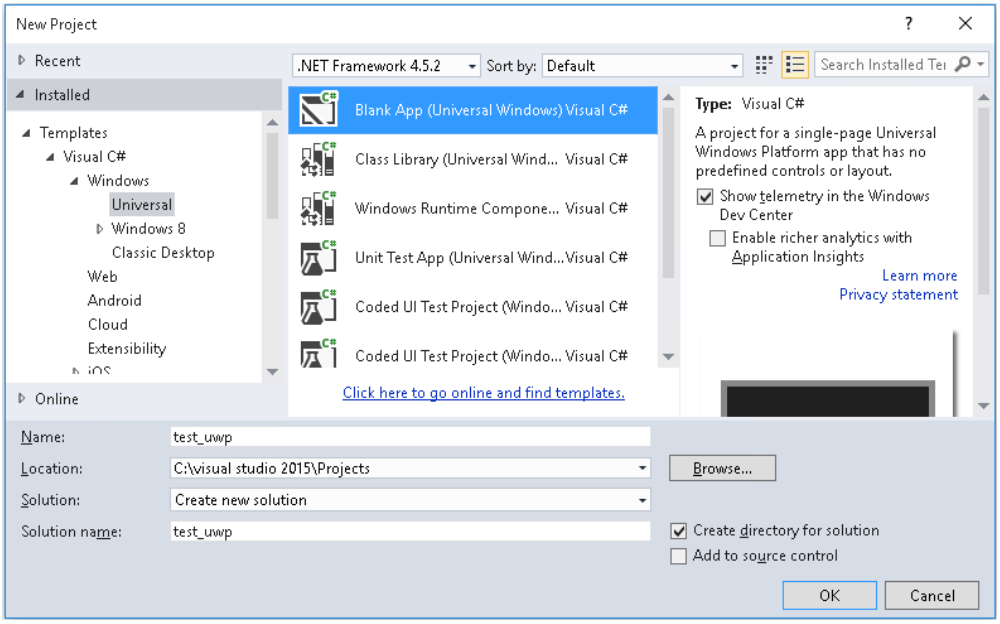

Figure 3-12

To run this project in Visual Studio 2015, please follow the steps below: (In this guide, we also use the local machine with x86 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 3-13.

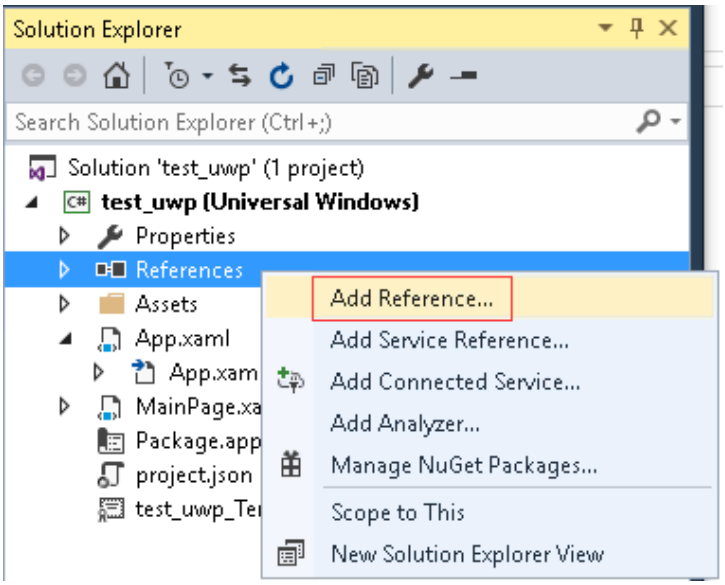

Figure 3-13

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 3-14.

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".*

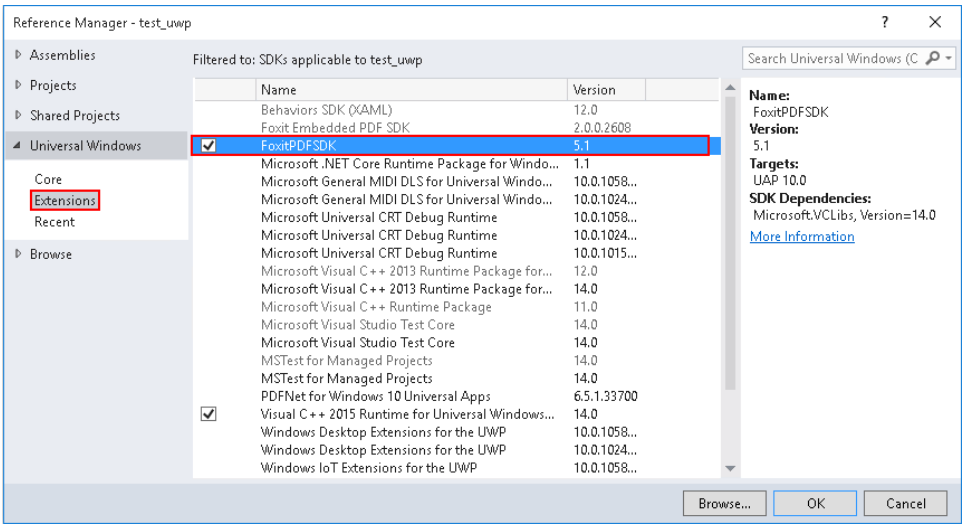

Figure 3-14

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 3-15.

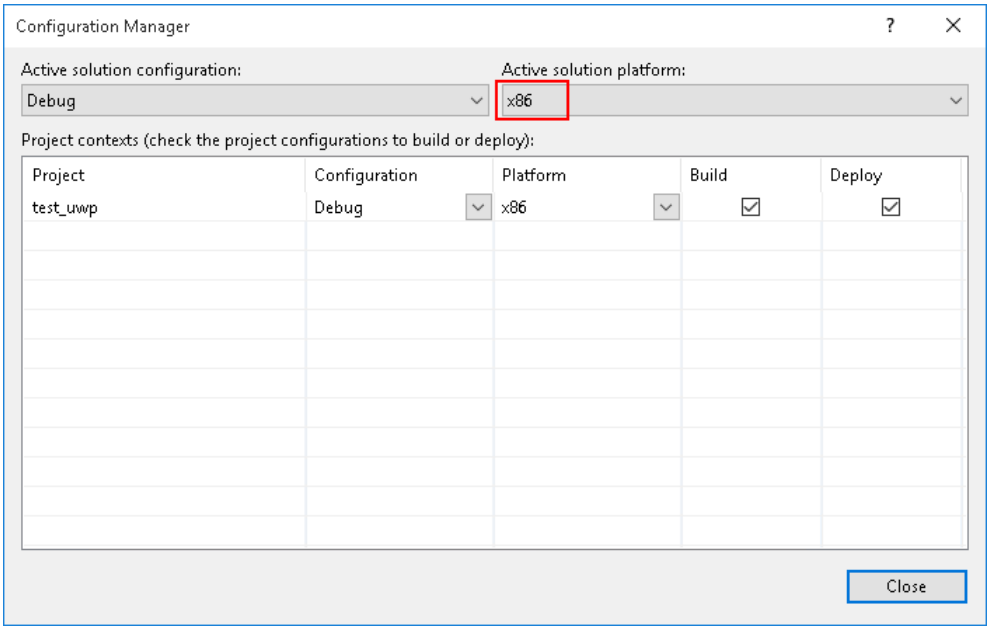

Figure 3-15

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) Load Foxit PDF SDK library. 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 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.
Size size = m_curPage.GetSize();
float m_PageWidth = (float)size.Width;
float m_PageHeight = (float)size.Height;

Foxit.PixelSource bitmap = new Foxit.PixelSource();
bitmap.Width = (int)m_PageWidth;
bitmap.Height = (int)m_PageHeight;
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 **Local Machine** to build and run the project. The screenshot of the project is shown in Figure 3-16.

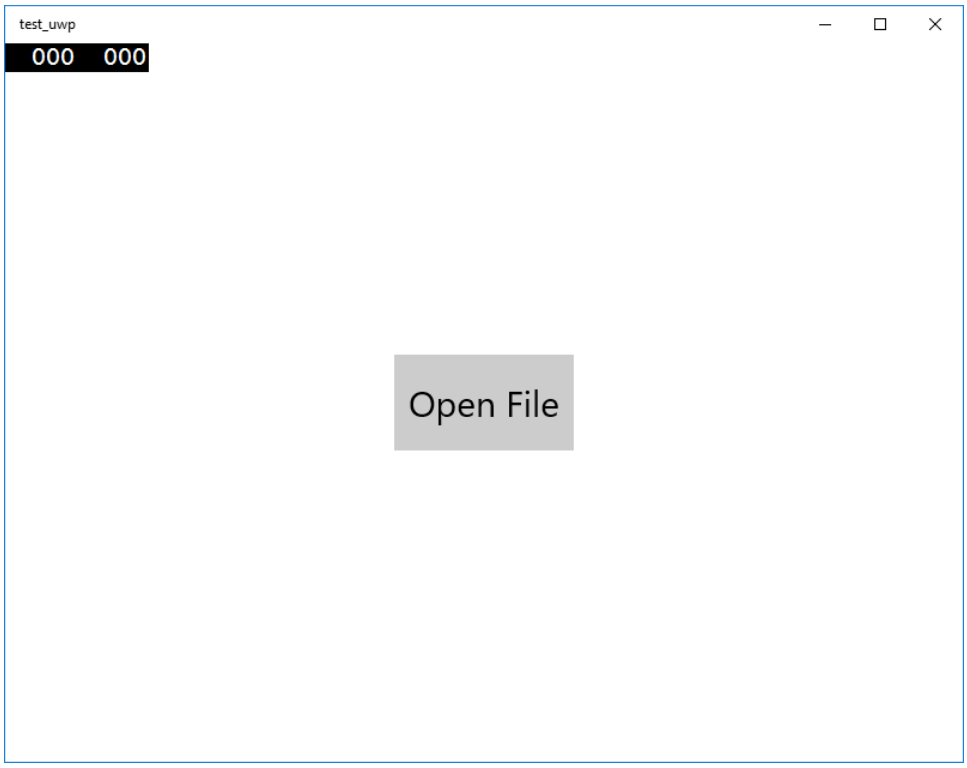

Figure 3-16

e) Click on **Open File** to choose and open a PDF document. A selecting window will be popped up as shown in Figure 3-17 when you click on **Open File.**

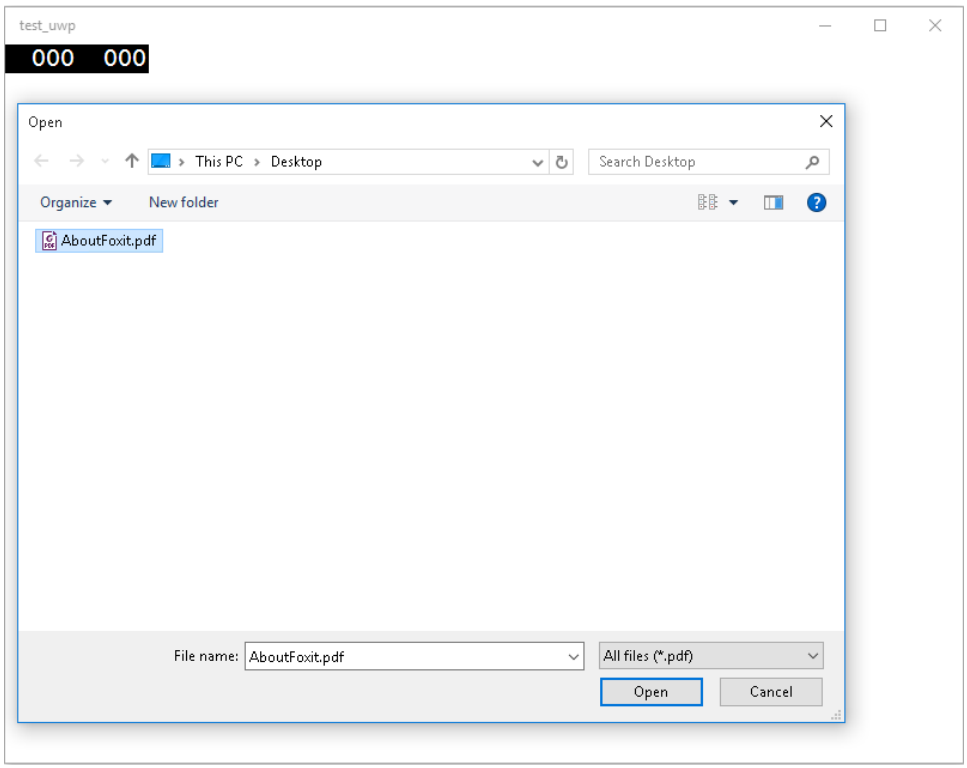

Figure 3-17

f) The selected PDF document will be displayed as shown in Figure 3-18.

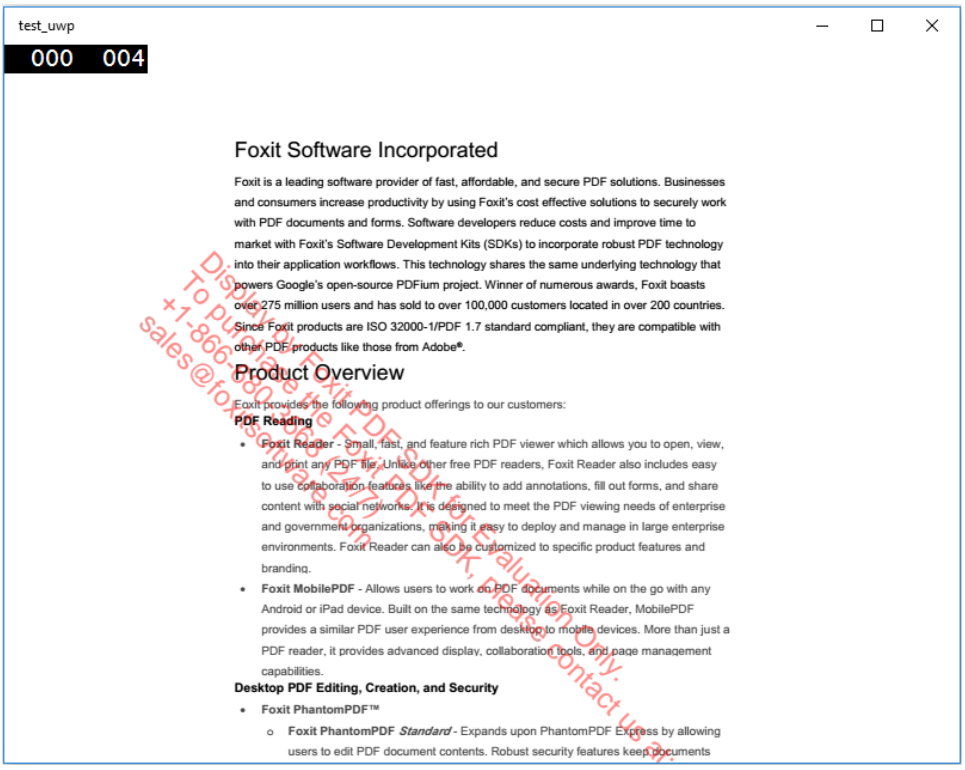

Figure 3-18

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();

            // Load 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.
            Size size = m_curPage.GetSize();
            float m_PageWidth = (float)size.Width;
            float m_PageHeight = (float)size.Height;

            // Render and display the current PDF page.
            Foxit.PixelSource bitmap = new Foxit.PixelSource();
            bitmap.Width = (int)m_PageWidth;
            bitmap.Height = (int)m_PageHeight;
            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;
            }
        }
    }
}
```

4 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 that target the Windows 10 Universal Apps. You can refer to the API reference [2] to get more details about the APIs used in all of the examples.

4.1 Load Library

It is necessary for applications to load Foxit PDF SDK before calling any APIs. The function Foxit::Library::Load is provided to load 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::Library::Unload to unload it.

Note *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=").*

Example:

4.1.1 How to load Foxit PDF SDK

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

4.2 Document

Document level APIs provide functions to open and close files, get page information, get the root bookmark, get a specific attachment and etc. PDF document can be loaded by function Foxit::PDF::Document::LoadAsync or Foxit::PDF::Document::LoadFromStreamAsync. After loading a PDF document, user can get a PDF page, get an attachment, access the PDF bookmarks and etc. Function PDF::Document::Close should be called to close the loaded PDF document when there is no need to access it any more.

Example:

4.2.1 How to load a PDF document

```
//load a PDF document with LoadAsync
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.FileTypeFilter.Add(".pdf");

StorageFile pdfFile = null;
pdfFile = await openPicker.PickSingleFileAsync();

Foxit.PDF.Document document = new Foxit.PDF.Document();
await document.LoadAsync(pdfFile, password, 0);

ErrorCode ret = Foxit.Library.GetLastError();
if (ret == ErrorCode.Success)
{
      //close a PDF document
     document.Close();
}
```

```
//load a PDF document with LoadFromStreamAsync
FileOpenPicker openPicker = new FileOpenPicker();
openPicker.FileTypeFilter.Add(".pdf");

StorageFile pdfFile = null;
pdfFile = await openPicker.PickSingleFileAsync();

IRandomAccessStream pdfStream = null;
pdfStream = await pdfFile.OpenAsync(FileAccessMode.ReadWrite);

Foxit.PDF.Document document = new Foxit.PDF.Document();
await document.LoadFromStreamAsync(pdfStream, password, 0);

ErrorCode ret = Foxit.Library.GetLastError();
if (ret == ErrorCode.Success)
{
       //close a PDF document
       document.Close();
}
```

4.2.2 How to get a PDF page

```
//Load a PDF page
//Assuming Document document has been loaded.
int count = document.CountPages();

Page page = await document.LoadPageAsync(pageIndex, 0, null);

ErrorCode ret = Foxit.Library.GetLastError();
if (ret != ErrorCode.Success)
{
       document.Close();
       return;
}
//do something about page
...

page.Close();
document.Close();
```

4.2.3 How to save a PDF to a file

```
//Save a PDF to file
//Assuming Document document has been loaded.

await document.SaveAsAsync(storageSaveFile, (uint)SaveFlags.Incremental, null);

ErrorCode ret = Foxit.Library.GetLastError();
if (ret!= ErrorCode.Success)
{
     document.Close();
     return;
}
//do something else
...

document.Close();
```

4.3 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 application APIs to access attachments such as getting attachments, counting attachments, creating attachments and accessing properties of attachments.

Example:

4.3.1 How to insert an attachment file into a PDF

```
//Create and insert an embedded attachment file into a PDF
//Assuming Document document has been loaded.
...
Attachment attachment = await document.CreateAttachment(attachFile,
insertIndex);

ErrorCode ret = Foxit.Library.GetLastError();
if (ret != ErrorCode.Success || attachment == null)
{
      document.Close();
      return;
}
//do something about attachment
...
document.Close();
```

4.3.2 How to get and count the attachments in a PDF

```
//Get and Count the attachments in a PDF
//Assuming Document document has been loaded.
...
int count = document.CountAttachment();

ErrorCode ret = Foxit.Library.GetLastError();
if (ret != ErrorCode.Success || count <= 0)
{
      document.Close();
      return;
}

Attachment attachment = document.GetAttachment(0);

ret = Foxit.Library.GetLastError();
if (ret != ErrorCode.Success || attachment == null)
{
      document.Close();
      return;
}

//do something about attachment
...

document.Close();
```

4.4 Page

Pages are the basic and important component of PDF Document. Foxit PDF SDK provide APIs to render the PDF page, get some basic information of PDF page, and retrieve a text page. PDF page can be loaded by function PDF::Document::LoadPageAsync, and it should be closed by calling function PDF::Page::Close when there is no need to access it any more.

Example:

4.4.1 How to get the size of the PDF page

```
//Get the size of the PDF page
//Assuming Document document and Page page have been loaded.
...
Size size = page.GetSize();

ErrorCode ret = Foxit.Library.GetLastError();
if (ret != ErrorCode.Success)
{
      page.Close()
      document.Close();
      return;
}
//do something about size.Width and size.Height
...
document.Close();
```

4.5 Text Page

Text page contains all the text content of a PDF page and can be used for text related operation, such as retrieving characters data, text search and selection, getting URL formatted texts, and etc. Prior to text processing, user should first call function PDF::Page::LoadTextPage to load the textPage object.

Example:

4.5.1 How to retrieve characters data

```
//Retrieve characters data
//Assuming Document document and Page page have been loaded.
...
TextPage textPage = page.LoadTextPage(0);

ErrorCode ret = Foxit.Library.GetLastError();
if (ret != ErrorCode.Success || textPage == null)
{
      page.Close()
      document.Close();
      return;
}

int count = textPage.CountChars();

ret = Foxit.Library.GetLastError();
if (ret != ErrorCode.Success || count <= 0)
{
       textPage.Release();
       page.Close()
       document.Close();
       return;
}

CharRange charRange = new CharRange(0, count);
String strText = textPage.GetChars(charRange);

ret = Foxit.Library.GetLastError();
if (ret != ErrorCode.Success)
{
       textPage.Release();
       page.Close();
       document.Close();
       return;
}
//do something else
...

textPage.Release();
page.Close();
document.Close();
```

4.5.2 How to search text in a PDF document

```
//Search text in a PDF document
//Assuming Document document, Page page and TextPage textPage have been loaded.
...

TextSearch search = null;
//whole word is compared with no case sensitive
search = textPage.StartSearch("foxit", SearchFlag.MatchWholeWord, 0);
boolean next = search.FindNext();

if(!next) return true;

//A match is found here
TextSelection select = search.GetSelection();
int rectnum = select.CountPieces();

for(int i = 0; i < rectnum; i++)
{
	RectF rectF = select.GetPieceRect(i);
	...
}

select.Release();
search.Release();
```

4.5.3 How to select text in a PDF document

```
//Select text in a PDF document
//Assuming Document document, Page page and TextPage textPage have been loaded.
...
CharRange charRange = new CharRange(0, -1);
TextSelection selection = textPage.SelectByRange(charRange);

string s = selection.GetChars();

selection.Release();
```

4.5.4 How to get the first URL formatted texts in a PDF page

```
//Get the first URL formatted text in a PDF page if there's any.
//Assuming Document document, Page page and TextPage textPage have been loaded.
...
int iLinkCount = textPage.CountLinks();
if (iLinkCount>0)
	String linkString = textPage.GetLink(0);
```

4.6 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::Document::GetBookmarkRoot 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 called function PDF::Bookmark::GetFirstChild.

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

- Function PDF::Document::FindBookmark can be called to try to find a bookmark with specific title.
- Function PDF::Bookmark::GetFirstChild can be called to get the first child bookmark of a bookmark if exists.
- Function PDF::Bookmark::GetNextSibling can be called to get the next sibling bookmark of a bookmark if exists.

Example:

4.6.1 How to find and list all bookmarks of a PDF

```
//Find and list all bookmarks of a PDF
//Assuming Document document has been loaded.
...
Bookmark bookmarkRoot = document.GetBookmarkRoot();
ErrorCode ret = Foxit.Library.GetLastError();
if (ret == ErrorCode.Success)
{
	Bookmark bookmarkFirstChild = bookmarkRoot.GetFirstChild();
	if (bookmarkFirstChild != null)
	{
		TraversalBookmark(bookmarkFirstChild, 0);
	}
}

private void TraversalBookmark(Bookmark rootBm, int iLevel)
{
	if (rootBm != null)
	{
		Bookmark child = rootBm.GetFirstChild();
		while (child != null)
		{
		   TraversalBookmark(child, iLevel + 1);
		   child = child.GetNextSibling();
		}
	 }
}
```

4.7 Annotations

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 4-1. 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 4-1 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 4-1

Annotation type	Description	Markup	Supported by SDK
Text(Note)	Text annotation	Yes	Yes
Link	Link Annotations	No	Yes
FreeText(TypeWritter)	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
Ink(pencil)	Ink annotation	Yes	Yes
Popup	Popup annotation	Yes	Yes
File Attachment	FileAttachment annotation	Yes	Yes
Sound	Sound annotation	Yes	No
Movie	Movie annotation	No	No
Screen	Screen annotation	Yes	No
PrinterMark	PrinterMark annotation	No	No
TrapNet	Trap network annotation	No	No
3D	3D annotation	Yes	No

Note: 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:

4.7.1 How to add a link annotation with a URI action.

```
//Add a link annotation with a URI action
...

RectF rect = new RectF(50, 200, 150, 150);
Link newLinkAnnot = (Link)pdfPage.AddAnnot(rect, AnnotType.Link, null, 0);

// Set some properties of the new link annot.
newLinkAnnot.SetContents("New link annot");
newLinkAnnot.SetFlags((uint)Flags.Print);
BorderInfo border = new BorderInfo(0xff0000ff, 2, BorderStyle.Solid, 0, 0,
null);
newLinkAnnot.SetBorderInfo(border);
...
// Set action to the new link annot
Foxit.PDF.Action action = new Foxit.PDF.Action();
action.Type = ActionType.Uri;
URIAction uriActData = new URIAction("http://www.foxitsoftware.com", false);
action.actionData = uriActData;
newLinkAnnot.InsertAction(ActionTrigger.MouseButtonUp, 0, action));
// Reset the appearance stream.
newLinkAnnot.ResetApppearanceStream();
```

4.7.2 How to add a highlight annotation over first 5 characters in a PDF page

```
// Add a highlight annot over first 5 characters in a PDF page.
// Assume the PDF document, PDF page and text page have all been loaded
successfully.
...


// Add a new highlight with empty rectangle and later Link::SetQuadPoints()
must be called.
RectF rect = new RectF(0, 0, 0, 0);
Highlight newHighlightAnnot = (Highlight)pdfPage.AddAnnot(rect,
AnnotType.Highlight, null, 0);

// Set some properties of the new link annot.
newHighlightAnnot.SetContents("New highlight annot");
newHighlightAnnot.SetFlags((uint)Flags.Print);
BorderInfo border = new BorderInfo(0xff00ff00, 2, BorderStyle.Solid, 0, 0,
null);
newHighlightAnnot.SetBorderInfo(border);
...
// Prepare the quadpoints and set to new highlight annot.
TextSelection selection = textPage.SelectByRange(charRange);
int iPieceCount = selection.CountPieces();
List<PointF> quadPoints = new List<PointF>();
for (int i = 0, j = 0; i < iPieceCount; i++)
{
    RectF rect = selection.GetPieceRect(i);
    if (null != rect)
    {
        quadPoints.Add(new PointF(rect.Left, rect.Top));
        quadPoints.Add(new PointF(rect.Right, rect.Top));
        quadPoints.Add(new PointF(rect.Left, rect.Bottom));
        quadPoints.Add(new PointF(rect.Right, rect.Bottom));
    }
}
newHighlightAnnot.SetQuadPoints(quadPoints.ToArray());
// Reset appearance stream.
newHighlightAnnot.ResetApppearanceStream();
```

4.8 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: *The Signature module only provides the third-party signature interface and requires the customers have their own signature implementation. If you want to purchase Foxit PDF SDK license and use any functions of this module, please contact Foxit to enable this module explicitly.*

Example:

4.8.1 How to add a signature and sign it.

```
// Add a signature and sign it
...

// Add a signature
RectF rect = new RectF(50, 150, 150, 50);
Signature sig = pdfDoc.AddSignature(pdfPage, rect, 8196);
// Set some information if necessary.
sig.SetLocation("Foxit");
sig.SetReason("For Test");
sig.SetSigner("Foxit");
// Set annot flags.
sig.SetAnnotFlags((uint)Foxit.PDF.Annotations.Flags.Print);
// Set appearance flags.
uint apFlags = (uint)(SigAppearanceFlags.FoxitFlag |
SigAppearanceFlags.CreationTime | SigAppearanceFlags.Signer |
SigAppearanceFlags.Reason | SigAppearanceFlags.Label);
sig.SetAppearanceFlags(apFlags);
// Reset appearance stream.
sig.ResetAppearanceStream();
// Set preferred filter and subfilter name.
String filter = "Adobe.PPKLite";
sig.SetPreferredFilter(filter);
String subfilter = "adbe.pkcs7.detached";
sig.SetPreferredSubFilter(subfilter);
// Register signature handler and sign the signature.
MySignatureHandler mySigHandler = new MySignatureHandler();
Library.RegisterSignatureHandler(filter, subFilter,
mySigHandler.m_sigHandler);

// "saveFolderPath" represents the path to the folder where the signing
result will be saved.
StorageFolder saveFolder = await
StorageFolder.GetFolderFromPathAsync(saveFolderPath);
if (null != saveFolder)
{
       StorageFile saveFile = await
saveFolder.CreateFileAsync("SignedResult.pdf",
CreationCollisionOption.ReplaceExisting);
       if (null != saveFile)
       {
       await sig.SignAsync(saveFile, null);
       }
}
```

Implement signature callback function of signing on MySignatureHandler class

```
class MySignatureHandler
{
       public SignatureHandler m_sigHandler = null;

        public MySignatureHandler()
        {
            m_sigHandler = new SignatureHandler();

            m_sigHandler.StartCalcDigestFunc += StartCalcDigestImp;
            m_sigHandler.ContinueCalcDigestFunc += ContinueCalcDigestImp;
            m_sigHandler.FinishCalcDigestFunc += FinishCalcDigestImp;
            m_sigHandler.SignFunc += SignImp;
            m_sigHandler.VerifyFunc += VerifyImp;
        }

        public Object StartCalcDigestImp(Object clientData, Signature sig,
IBuffer fileBuffer, uint[] byteRangeArray)
        {
              //TODO: implemented by user.
              return null;
        }

        public int ContinueCalcDigestImp(Object clientData, Object context,
Pause pause)
        {
              //TODO: implemented by user.
              return 0;
        }

        public IBuffer FinishCalcDigestImp(Object clientData, Object context)
        {
              //TODO: implemented by user.
              return null;
        }

        public IBuffer SignImp(Object clientData, Object context, Signature
sig, IBuffer digest)
        {
              //TODO: implemented by user.
              return null;
        }

        public bool VerifyImp(Object clientData, Object context, Signature sig,
IBuffer digest, IBuffer signedData)
        {
              //TODO: implemented by user.
              return false;
        }
    }
```

4.9 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:

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

```
// Assuming that we've got a link annot.
...

Foxit.PDF.Action action = new Foxit.PDF.Action();
action.Type = ActionType.Uri;
URIAction uriActData = new URIAction("http://www.foxitsoftware.com", false);
action.actionData = uriActData;
linkAnnot.InsertAction(ActionTrigger.MouseButtonUp, 0, action);
```

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

```
// Assuming that we've got a link annot.
...

Foxit.PDF.Action action = new Foxit.PDF.Action();
action.Type = ActionType.Goto;
Destination dest = new Destination(0, ZoomMode.FitPage, null);
GotoAction gotoAction = new GotoAction(dest);
action.actionData = gotoAction;
linkAnnot.InsertAction(ActionTrigger.MouseButtonUp, 0, action);
```

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