DEVELOPER GUIDE

Foxit PDF SDK

Windows Universal Platform on Windows 10

Microsoft® Partner

Gold Independent Software Vendor (ISV)

TABLE OF CONTENTS

1 Introduction to Foxit PDF SDK

1.1 Foxit PDF SDK

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.

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 UWP platform.

1.2 Foxit PDF SDK for UWP

Have you ever worried about the complexity of the PDF specification? Or have you ever felt lost when asked to build a full-featured PDF app within a limited time-frame? If your answer is "Yes", then congratulations! You have just found the best solution in the industry for rapidly integrating PDF functionality into your apps.

Foxit PDF SDK for UWP focuses on helping developers easily integrate powerful Foxit PDF technology into their own apps. With this SDK, even developers with a limited knowledge of PDF can quickly build a professional PDF viewer with just a few lines of code on Universal Windows Platform (UWP) for Windows 10.

1.2.1 Why Foxit PDF SDK for UWP is your choice

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

Foxit PDF SDK for UWP provides quick PDF viewing and manipulation support for Windows 10 Universal Windows Platform. Customers choose it for the following reasons:

- **Easy to integrate**

Developers can seamlessly integrate the SDK into their own apps with just a few lines of code.

- **Perfectly designed**

Foxit PDF SDK for UWP is designed with a simple, clean, and friendly style, which provides the best user experience.

- **Flexible customization**

Foxit PDF SDK for UWP provides the source code for the user interface which lets the developers have full control of the functionality and appearance of their apps.

- **Robust performance on mobile platforms**

Foxit PDF SDK for UWP provides an OOM (out-of-memory) recovery mechanism to ensure the app has high robust performance when running the app on a mobile device which offers limited memory.

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

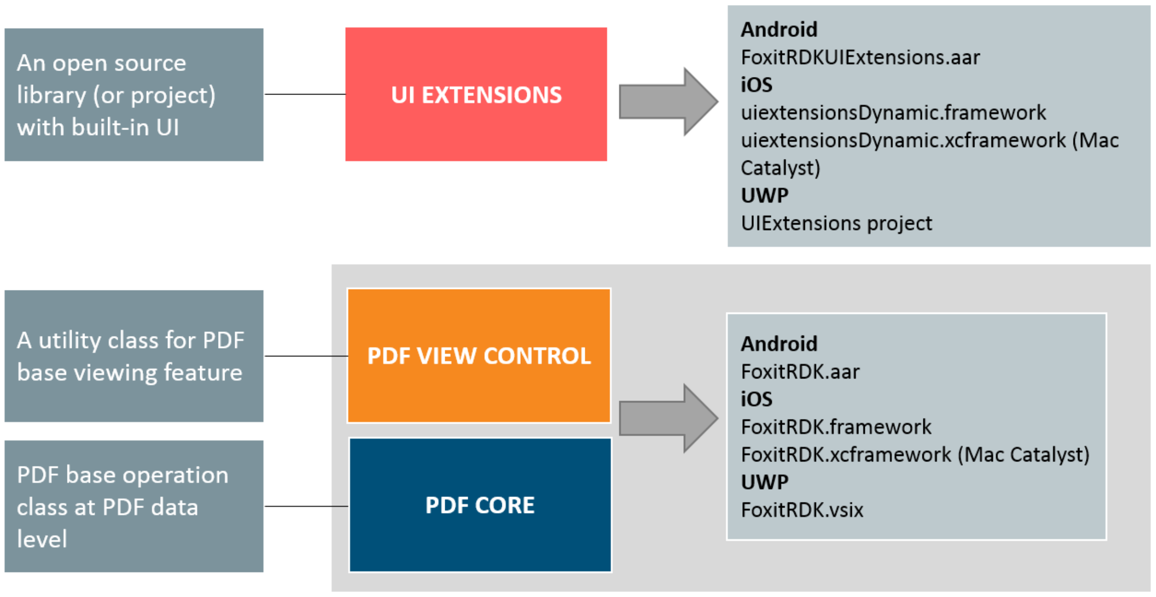
The core technology of the SDK is based on Foxit's PDF engine, which is trusted by a large number of the world's largest and well-known companies. Foxit's powerful engine makes the app fast on parsing, rendering, and makes document viewing consistent on a variety of devices.

- **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.

1.2.2 Main Frame of Foxit PDF SDK for UWP

Foxit PDF SDK for UWP consists of three elements as shown in the following picture. This structure is shared between all mobile platform versions of Foxit PDF SDK, which makes it easier to integrate and support multiple mobile operating systems and frameworks in your apps.


An open source library (or project) with built-in UI
UI EXTENSIONS
Android
FoxitRDKUIExtensions.aar
iOS
uiextensionsDynamic.framework
uiextensionsDynamic.xcframework (Mac Catalyst)
UWP
UIExtensions project
A utility class for PDF base viewing feature
PDF base operation class at PDF data level
PDF VIEW CONTROL
PDF CORE
Android
FoxitRDK.aar
iOS
FoxitRDK.framework
FoxitRDK.xcframework (Mac Catalyst)
UWP
FoxitRDK.vsix

The three elements of Foxit PDF SDK for Android, iOS and UWP

- **PDF Core API**

The PDF Core API is the heart of this SDK and is built on Foxit's powerful underlying technology. It provides the functionality for basic PDF operation features, and is utilized by the PDF View Control and UI Extensions project, which ensures the apps can achieve high performance and efficiency. The Core API can be used independently for document rendering, analysis, text extraction, text search, form filling, annotation creation and manipulation and much more.

- **PDF View Control**

The PDF View Control is a utility class that provides the functionality for developers to interact with rendering PDF documents per their requirements. With Foxit's renowned and widely used PDF rendering technology at its core, the View Control provides fast and high-quality rendering, zooming, scrolling and page navigation features. The View Control derives from platform related viewer classes and allows for extension to accommodate specific user needs.

- **UI Extensions Project**

The UI Extensions Project is an open source project that provides a customizable user interface with built-in support for text selection, markup annotation, outline navigation, form filling, text reflow, and document editing. These features in the UI Extensions are implemented using the PDF Core API and PDF View Control. Developers can utilize these ready-to-use UI implementations to build a PDF viewer quickly with the added benefit of complete flexibility and control to customize the UI design as desired.

1.2.3 Key Features of Foxit PDF SDK for UWP

Foxit PDF SDK for UWP has several main features which help app developers quickly implement the functions that they really need and reduce the development cost.

Features	
PDF Document	Open and close files, set and get metadata.
PDF Page	Parse, render, read, and edit PDF pages.
Render	Graphics engine created on a bitmap for platform graphics device.
Reflow	Rearrange page content.
Text Select	Search text in a PDF document
Text Search	Search text in a PDF document.
Bookmark(Outline)	Directly locate and link to point of interest within a document.
Annotation	Create, edit and remove annotations.
Form	Fill form with JavaScript support, export and import form data by XFDF/FDF/XML file.
Out of Memory	Recover from an OOM condition.

***Note** Outline is the technical term used in the PDF specification for what is commonly known as bookmarks in traditional desktop PDF viewers. Reading bookmarks are commonly used on mobile and tablet PDF viewers to mark progress and interesting passages as users read but are not technically outline and are stored at app level rather than within the PDF itself.*

1.3 Evaluation

Foxit PDF SDK allows users to download trial version to evaluate SDK. The trial version has no difference from the standard licensed version except for the free trial limitation and the trial watermarks in the generated pages. After the evaluation period expires, customers should contact the 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 developers permission to release their apps which utilize Foxit PDF SDK. However, users are prohibited to distribute any documents, sample code, or source code in the released package of Foxit PDF SDK to any third party without written permission from Foxit Software Incorporated.

1.5 About this Guide

This guide is intended for the developers who need to integrate Foxit PDF SDK for UWP into their own apps for both PC/tablet and Windows Phone. It aims at introducing the following sections:

- Section 1: gives an introduction of Foxit PDF SDK, especially for UWP SDK.
- Section 2: illustrates the package structure, running demo.
- Section 3: describes how to quickly create a full-featured PDF Reader.
- Section 4: provides support information.

2 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 it with Universal Windows Platform (UWP) apps on Windows 10. The following sections introduce the structure of the installation package, how to run a demo, and how to create your own project.

2.1 System Requirements

Windows 10 Universal Apps:

Windows 10

Recommend to install Visual Studio 2017 (with Universal Windows App Development Tools)

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

2.2 What is in the Package

Download the "foxitpdfsdk_7_0_2_uwp.zip" package, and extract it to a new directory like "foxitpdfsdk_7_0_2_uwp" as shown in Figure 2-1. The package contains:

docs:	A folder containing API references, developer guide.
libs:	A folder containing license files, FoxitRDK.vsix, and UI Extensions project.
samples:	A folder containing UWP sample projects.
FoxitSDK.sln	A solution file for the demos.
legal.txt:	Legal and copyright information.
release_notes.txt:	Release information.

***NOTE**: the highlighted rectangle in the figure is just the version of Foxit PDF SDK for UWP. Here the SDK version is 7.0.2, so it shows 7_0_2 in the package name. Other highlighted rectangles in other figures have the same meaning in this guide.*

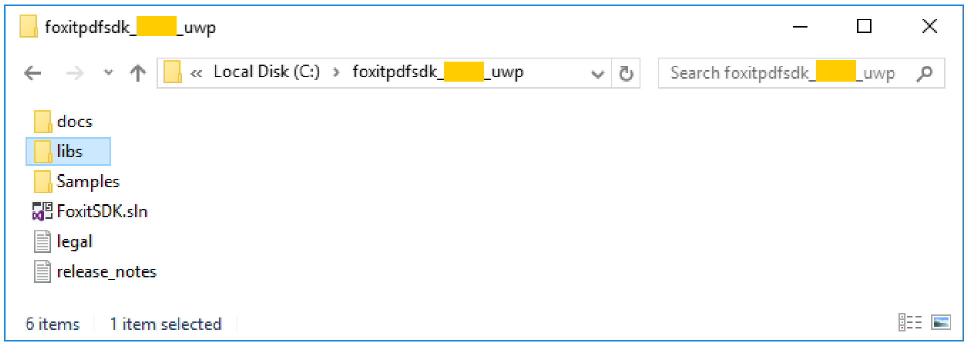

foxitpdfsdk_[redacted]_uwp
Local Disk (C:) › foxitpdfsdk_[redacted]_uwp
Search foxitpdfsdk_[redacted]_uwp
docs
libs
Samples
FoxitSDK.sln
legal
release_notes
6 items
1 item selected

Figure 2-1

In the "libs" folder as shown in Figure 2-2, there are items that make up the core components of Foxit PDF SDK for UWP.

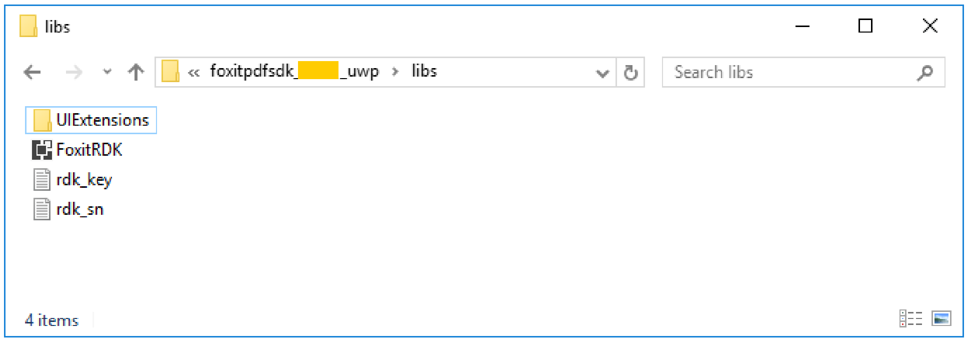


Figure 2-2

- ***FoxitRDK.vsix*** - The extension package.
- ***UIExtensions*** project - It is an open source project that contains the UI implementations for the basic UI of app and the ready-to-use feature module UI, which can help developers rapidly embed a fully functional PDF reader into their UWP app. Of course, developers are not forced to use the default UI, they can freely customize and design the UI for their specific apps through the "UIExtensions" project.

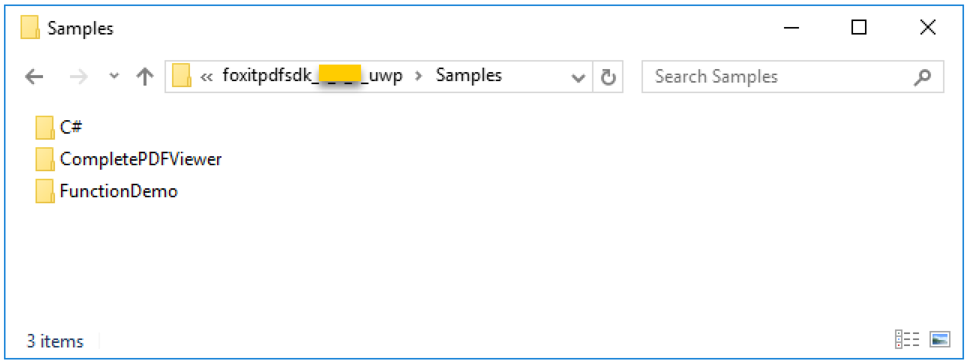
At this point you should just be getting a feel for what Foxit PDF SDK package looks like, we're going to cover everything in detail in a bit.

2.3 How to run a demo

Download and install Visual Studio IDE (https://www.visualstudio.com/).

Note*: In this guide, we will run the demos in Visual Studio 2017. This guide does not cover the installation of Visual Studio. You can refer to Visual Studio's developer site if you haven't installed it already.*

Foxit PDF SDK for Universal Windows Platform (UWP) on Windows 10 provides two useful C++ demos and one C# demo for developers to learn how to call the SDK as shown in Figure 2-3.



Samples

« foxitpdfsdk_[redacted]_uwp › Samples

Search Samples

C#
CompletePDFViewer
FunctionDemo

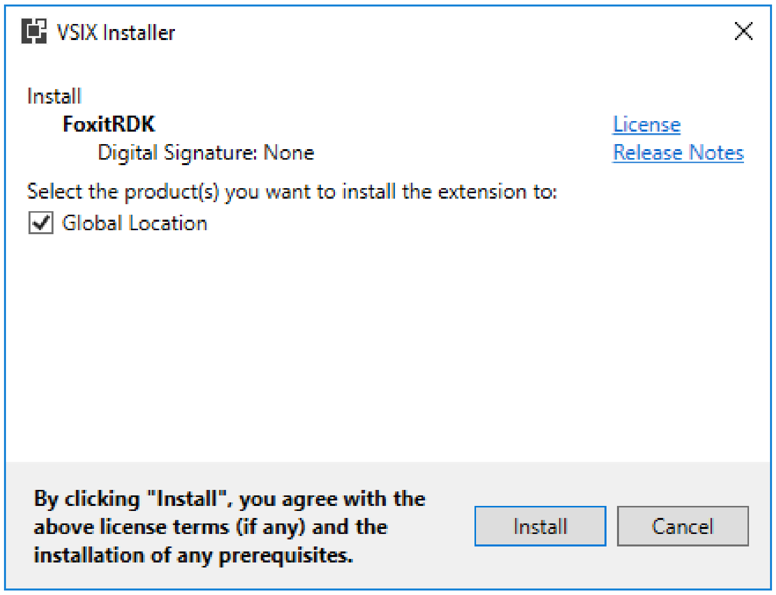
3 items

Figure 2-3

2.3.1 Install Foxit PDF SDK extension

To run the demos in Visual Studio 2017, Foxit PDF SDK extension should be installed at first.

Double-click **FoxitRDK.vsix** in the "lib" folder, click on **Install** in the "*VSIX Installer*" dialog to allow Foxit PDF SDK extension to be installed (see Figure 2-4). Note: Before installing Foxit PDF SDK extension, please close Visual Studio.



VSIX Installer

Install
FoxitRDK
Digital Signature: None

License
Release Notes

Select the product(s) you want to install the extension to:
[✓] Global Location

By clicking "Install", you agree with the above license terms (if any) and the installation of any prerequisites.

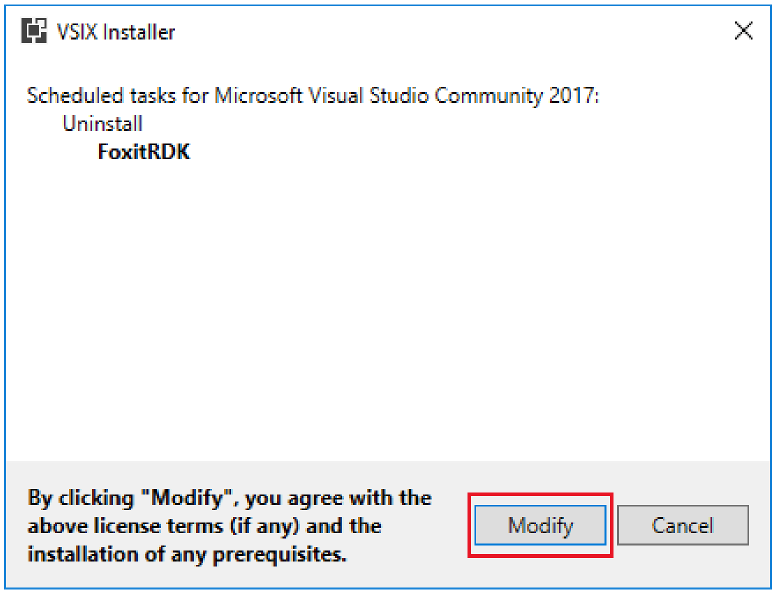
Install   Cancel

Figure 2-4

Note: *If you already have a version of Foxit PDF SDK for UWP installed, 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 2017 by following the steps:*

*a) Open menu "**Tools** -> **Extensions and Updates....**", under the menu **Installed** -> **All**, you should be able to find "**FoxitRDK**" extension. Select it and click **Uninstall**.*

*b) Close Visual Studio, then the "VSIX Installer" dialog will pop up as shown in Figure 2-5. Click **Modify** to uninstall the extension.*


VSIX Installer

Scheduled tasks for Microsoft Visual Studio Community 2017:
Uninstall
FoxitRDK

By clicking "Modify", you agree with the above license terms (if any) and the installation of any prerequisites.

Figure 2-5

2.3.2 Function demo

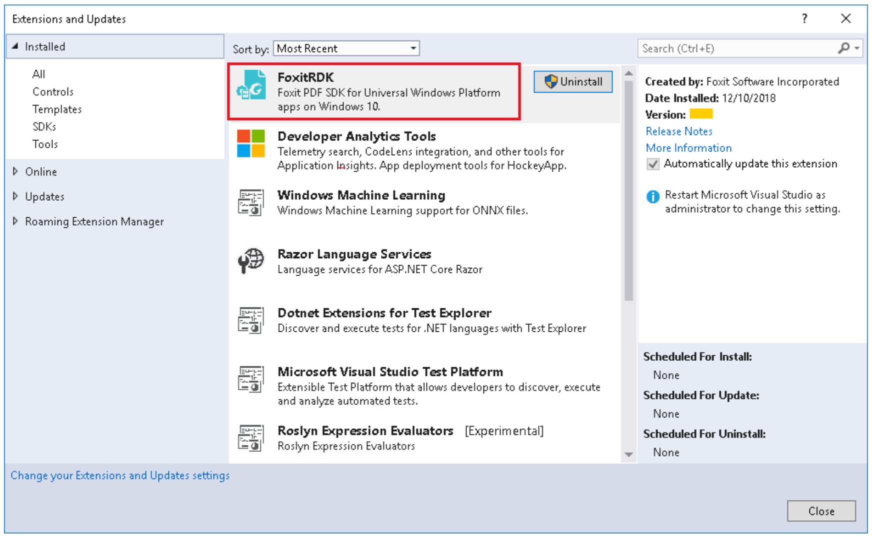
The function demo coded by C++ is provided to show how to use Foxit PDF SDK for UWP to realize some specific features related to PDF with PDF core API. This demo includes the following features:

- **pdf2txt**: extract text from a PDF document to a TXT file.
- **outline**: edit outline (aka bookmark) appearances and titles.
- **annotation**: add annotations to a PDF page.
- **docinfo**: export document information of a PDF to a TXT file.
- **render**: render a specified page to Bitmap.

To run it in Visual Studio 2017, follow the steps below: (in this guide, we use the local machine with x86 as an example to run the demos)

a) Refer to "Install Foxit PDF SDK extension" to install SDK extension.

b) Open **FoxitSDK.sln** in Visual Studio 2017. Then check whether 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 2-6.



Extensions and Updates

FoxitRDK
Foxit PDF SDK for Universal Windows Platform apps on Windows 10.

Developer Analytics Tools
Telemetry search, CodeLens integration, and other tools for Application Insights. App deployment tools for HockeyApp.

Windows Machine Learning
Windows Machine Learning support for ONNX files.

Razor Language Services
Language services for ASP.NET Core Razor

Dotnet Extensions for Test Explorer
Discover and execute tests for .NET languages with Test Explorer

Microsoft Visual Studio Test Platform
Extensible Test Platform that allows developers to discover, execute and analyze automated tests.

Roslyn Expression Evaluators [Experimental]
Roslyn Expression Evaluators

Created by: Foxit Software Incorporated
Date Installed: 12/10/2018
Version:
Release Notes
More Information
Automatically update this extension

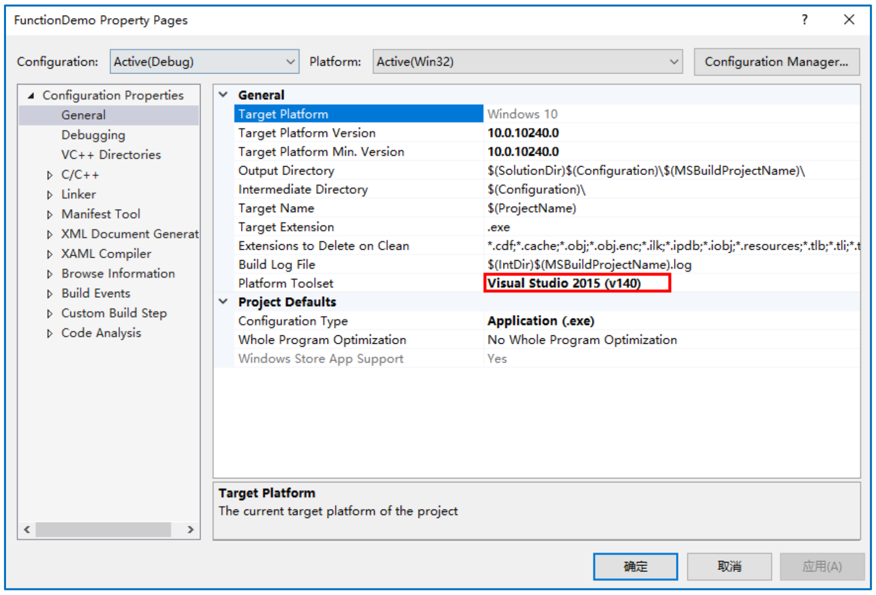
Restart Microsoft Visual Studio as administrator to change this setting.

Scheduled For Install:
None
Scheduled For Update:
None
Scheduled For Uninstall:
None

Change your Extensions and Updates settings

Figure 2-6

Note*: If you Open* ***FoxitSDK.sln*** *in Visual Studio 2015 to run the demos, you should change the* ***Platform Toolset*** *to Visual Studio 2015 (V140). Right-click the demo project in the solution explorer, select* ***Properties****. In the Property Pages dialog, select* ***General*** *->* ***Platform Toolset****, and choose Visual Studio 2015 (V140). (See* Figure 2-7*)*



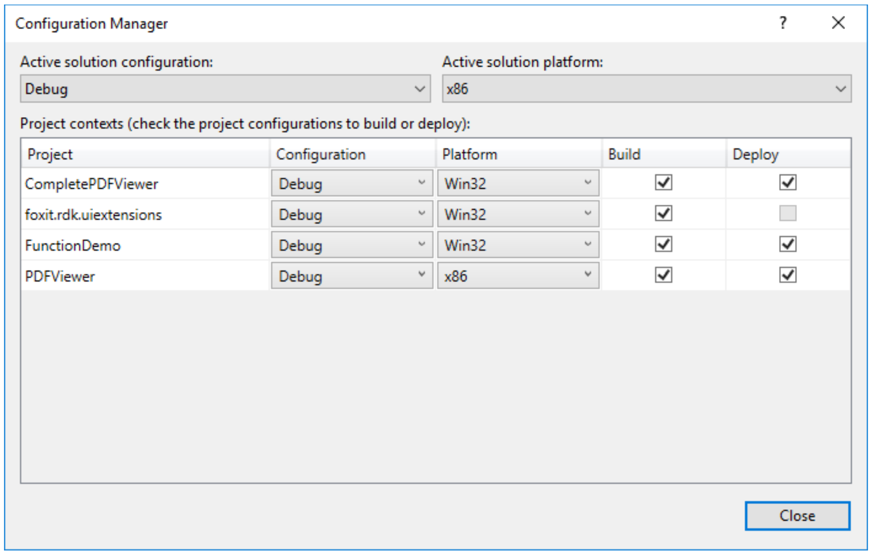
Configuration: Active(Debug) Platform: Active(Win32)

General	
Target Platform	Windows 10
Target Platform Version	**10.0.10240.0**
Target Platform Min. Version	**10.0.10240.0**
Output Directory	$(SolutionDir)$(Configuration)\$(MSBuildProjectName)\
Intermediate Directory	$(Configuration)\
Target Name	$(ProjectName)
Target Extension	.exe
Extensions to Delete on Clean	*.cdf;*.cache;*.obj;*.obj.enc;*.ilk;*.ipdb;*.iobj;*.resources;*.tlb;*.tli;*.t
Build Log File	$(IntDir)$(MSBuildProjectName).log
Platform Toolset	**Visual Studio 2015 (v140)**
Project Defaults	
Configuration Type	**Application (.exe)**
Whole Program Optimization	No Whole Program Optimization
Windows Store App Support	Yes

Target Platform
The current target platform of the project

Figure 2-7

c) Change the build architecture of the project. Click on **Build** -> **Configuration Manager** and choose **x86** for "Active solution platform" as shown in Figure 2-8.



Active solution configuration: Debug

Active solution platform: x86

Project contexts (check the project configurations to build or deploy):

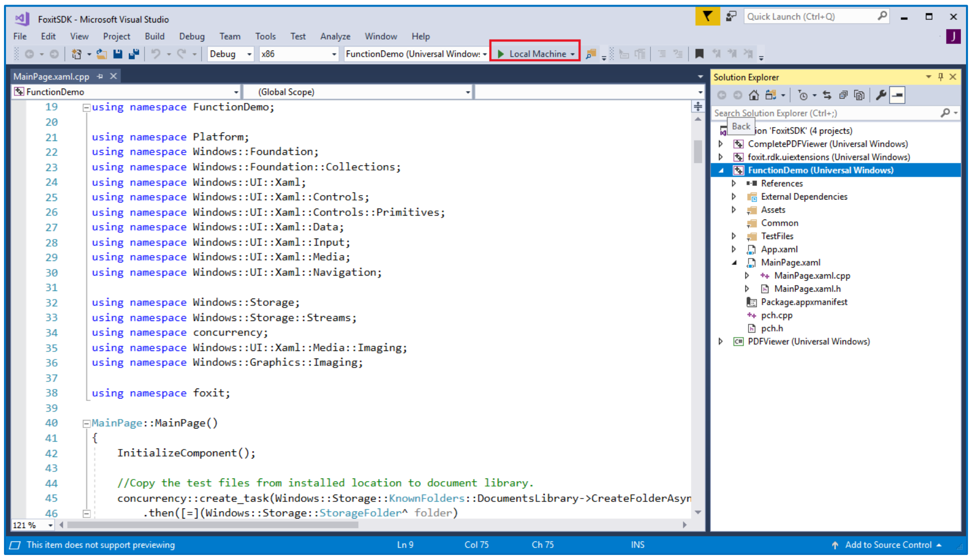
Project	Configuration	Platform	Build	Deploy
CompletePDFViewer	Debug	Win32	✓	✓
foxit.rdk.uiextensions	Debug	Win32	✓	
FunctionDemo	Debug	Win32	✓	✓
PDFViewer	Debug	x86	✓	✓

Figure 2-8

***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 "Active solution platform".*

d) Right-click the project "**FunctionDemo**" and select "Set as StartUp Project".

e) Click on **Local Machine** as shown in Figure 2-9 to build and run the demo. The screenshot of the demo is shown in Figure 2-10.


FoxitSDK - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Debug
x86
FunctionDemo (Universal Window:
Local Machine
Quick Launch (Ctrl+Q)
MainPage.xaml.cpp
FunctionDemo
(Global Scope)
121 %
This item does not support previewing
Ln 9
Col 75
Ch 75
INS
Add to Source Control
Solution Explorer
Search Solution Explorer (Ctrl+;)
Back

```
using namespace FunctionDemo;

using namespace Platform;
using namespace Windows::Foundation;
using namespace Windows::Foundation::Collections;
using namespace Windows::UI::Xaml;
using namespace Windows::UI::Xaml::Controls;
using namespace Windows::UI::Xaml::Controls::Primitives;
using namespace Windows::UI::Xaml::Data;
using namespace Windows::UI::Xaml::Input;
using namespace Windows::UI::Xaml::Media;
using namespace Windows::UI::Xaml::Navigation;

using namespace Windows::Storage;
using namespace Windows::Storage::Streams;
using namespace concurrency;
using namespace Windows::UI::Xaml::Media::Imaging;
using namespace Windows::Graphics::Imaging;

using namespace foxit;

MainPage::MainPage()
{
    InitializeComponent();

    //Copy the test files from installed location to document library.
    concurrency::create_task(Windows::Storage::KnownFolders::DocumentsLibrary->CreateFolderAsyn
        .then([=](Windows::Storage::StorageFolder^ folder)
```

```
ion 'FoxitSDK' (4 projects)
  CompletePDFViewer (Universal Windows)
  foxit.rdk.uiextensions (Universal Windows)
  FunctionDemo (Universal Windows)
    References
    External Dependencies
    Assets
    Common
    TestFiles
    App.xaml
    MainPage.xaml
      MainPage.xaml.cpp
      MainPage.xaml.h
    Package.appxmanifest
    pch.cpp
    pch.h
  PDFViewer (Universal Windows)
```

Figure 2-9

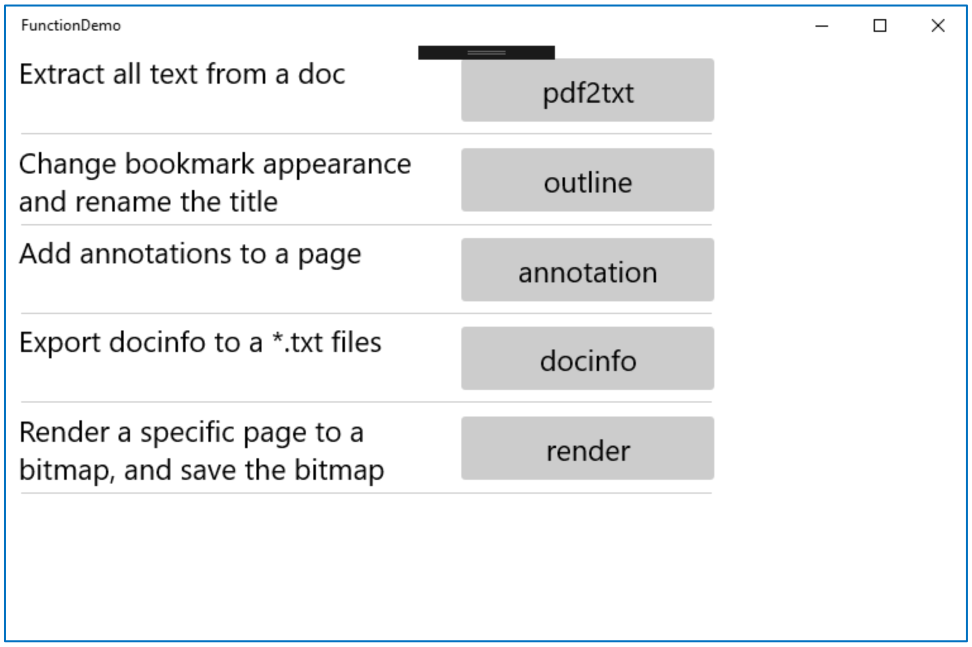
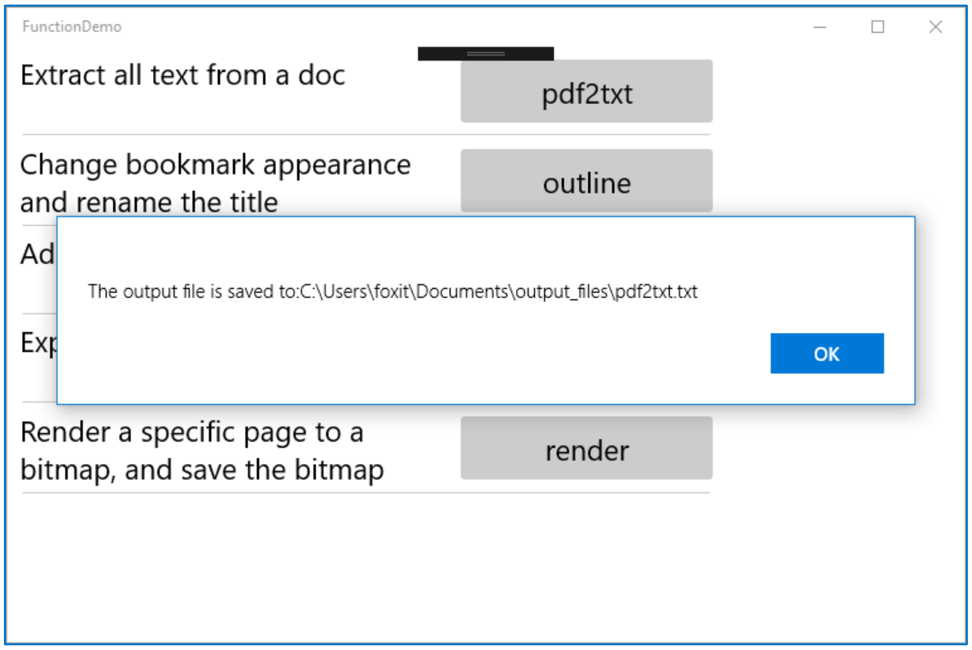

FunctionDemo
Extract all text from a doc
pdf2txt
Change bookmark appearance and rename the title
outline
Add annotations to a page
annotation
Export docinfo to a *.txt files
docinfo
Render a specific page to a bitmap, and save the bitmap
render

Figure 2-10

f) Click the feature buttons in the above picture to perform the corresponding actions. For example, click "pdf2txt", and then a message box will be popped up as shown in Figure 2-11.

It shows where the text file was saved to. Just run the demo and try the features.


FunctionDemo
Extract all text from a doc
pdf2txt
Change bookmark appearance and rename the title
outline
Ad
Exp
Render a specific page to a bitmap, and save the bitmap
render
OK

The output file is saved to:C:\Users\foxit\Documents\output_files\pdf2txt.txt

Figure 2-11

2.3.3 Complete PDF viewer demo

The complete PDF viewer demo coded by C++ demonstrates how to use Foxit PDF SDK for UWP to realize a completely full-featured PDF viewer which is almost ready-to-use as a real world mobile PDF reader. This demo utilizes all of the features and built-in UI implementations which are provided in Foxit PDF SDK.

To run the demo in Visual Studio 2017, please refer to the setup steps outlined in the Function demo.

Right-click the project "**CompletePDFViewer**" and select "Set as StartUp Project". Press F5 to build and run the demo. After building the demo successfully, on the start screen, it lists the "developer_guide_uwp.pdf" document (see Figure 2-12).

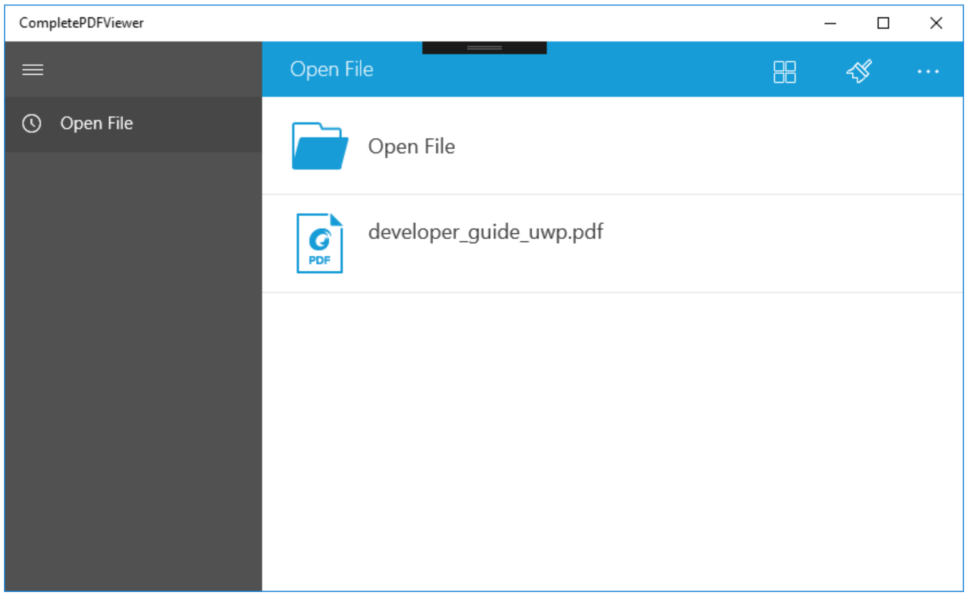


Figure 2-12

Click the "developer_guide_uwp.pdf" document, and then it will be opened and displayed as shown in Figure 2-13.

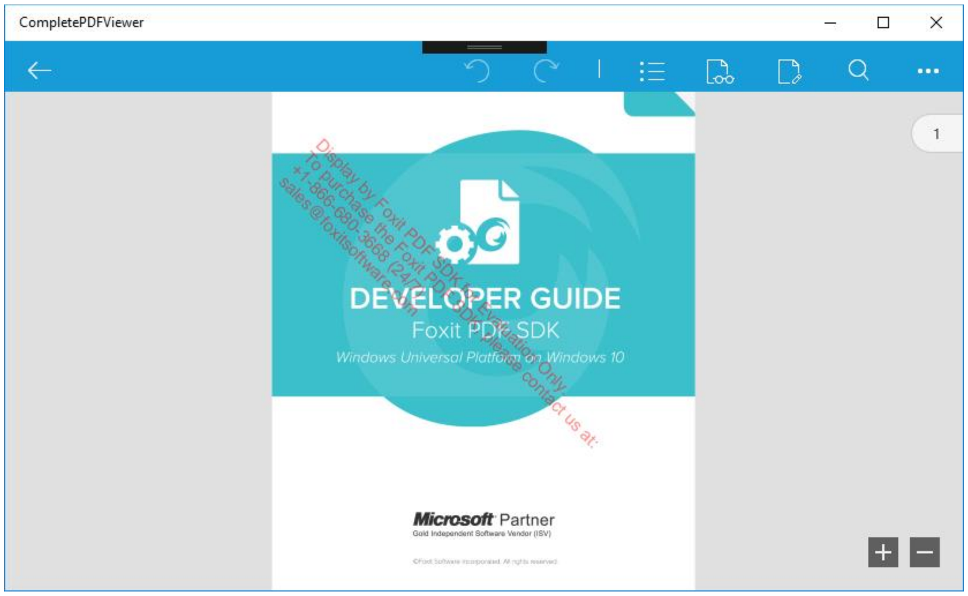


Figure 2-13

This demo realizes a completely full-featured PDF viewer, please feel free to run it and try it.

For example, it provides the page thumbnail feature. You can click at the top bar, choose the **Thumbnail** as shown in Figure 2-14, and then the thumbnail of the document will be displayed as shown in Figure 2-15.

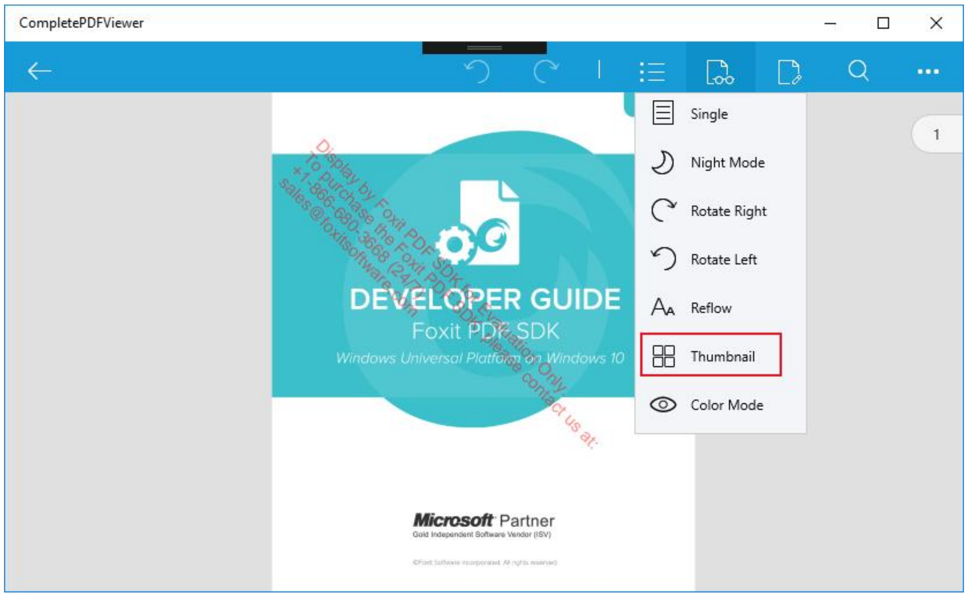


Figure 2-14

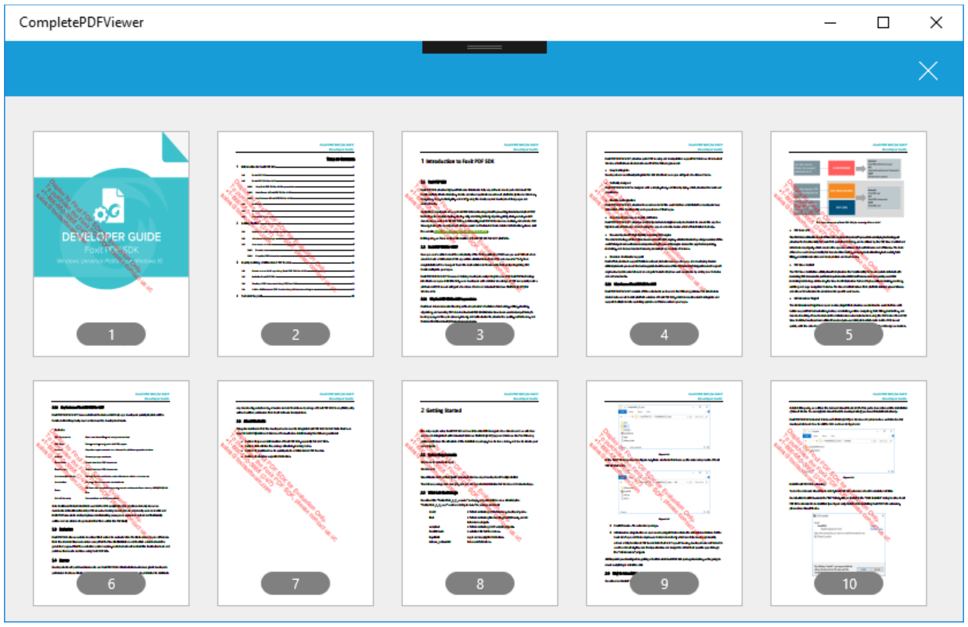


Figure 2-15

2.3.4 PDFViewer demo

The PDFViewer demo is similar to the Complete PDF viewer demo. It demonstrates how to use C# to implement a completely full-featured PDF viewer.

To run the demo in Visual Studio 2017, please refer to the setup steps outlined in the Function demo.

Right-click the project "**PDFViewer**" and select "Set as StartUp Project". Press F5 to build and run the demo. After building the demo successfully, on the start screen, Click the button "Open a PDF" (See Figure 2-16) to select a PDF document. For example, select the "developer_guide_uwp.pdf" in the "docs" folder of the download package, and then the document will be opened and displayed as shown in Figure 2-17. You can feel free to experience the full-featured PDF viewer.

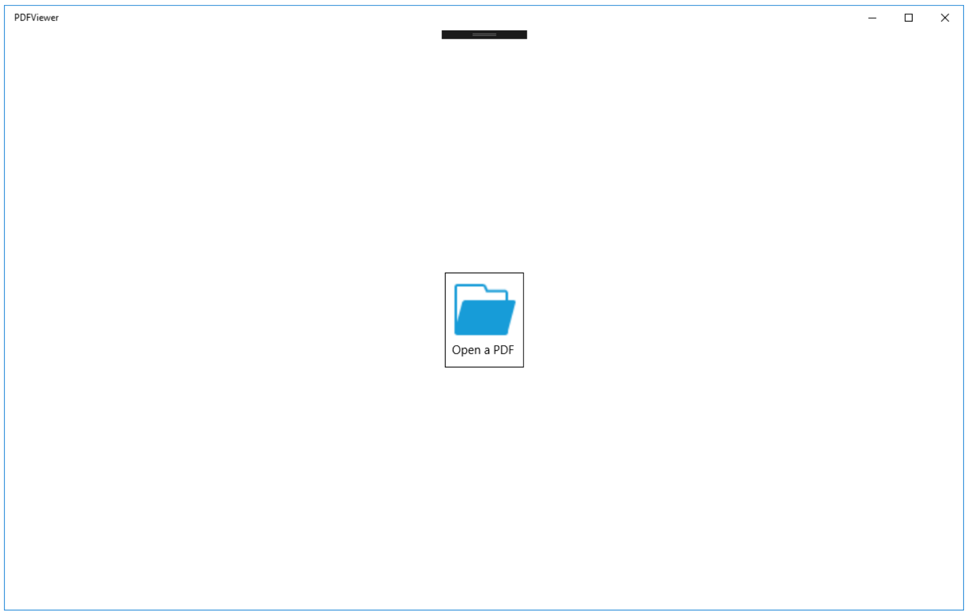


Figure 2-16

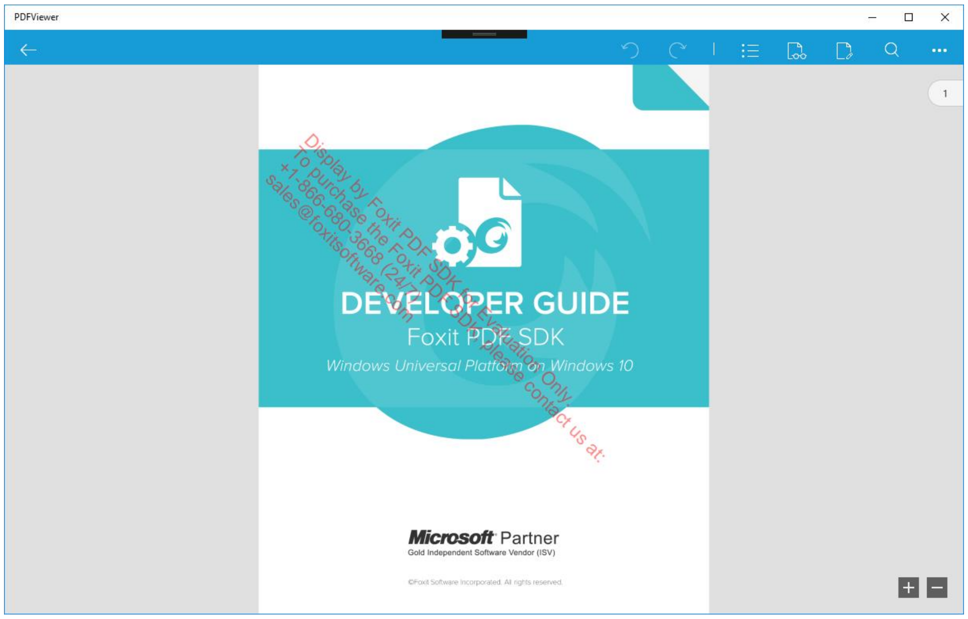

PDFViewer
1
DEVELOPER GUIDE
Foxit PDF SDK
Windows Universal Platform on Windows 10
Microsoft Partner
Gold Independent Software Vendor (ISV)
©Foxit Software Incorporated. All rights reserved.
Display by Foxit PDF SDK for Evaluation Only
To purchase the Foxit PDF SDK, please contact us at:
+1-866-680-3668 (24/7)
sales@foxitsoftware.com

Figure 2-17

3 Rapidly building a full-featured PDF Reader

Foxit PDF SDK for UWP wrapped all of the UI implementations including the basic UI for app and ready-to-use UI feature modules to UI Extensions project, so that developers can easily and rapidly build a full-featured PDF Reader with just a few lines of code. This section will help you to quickly get started with using Foxit PDF SDK for UWP to make a full-featured PDF Reader app in C++ and C# with step-by-step instructions provided. It includes the following steps:

3.1 Make a UWP app in C++ with Foxit PDF SDK for UWP

This section will help you to quickly make a UWP app in C++ using Foxit PDF SDK. It includes the following steps:

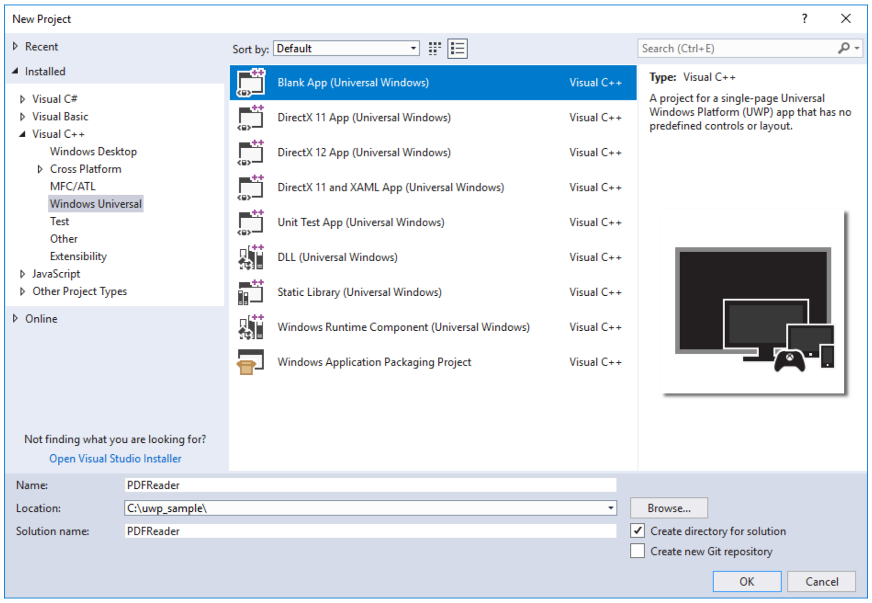
- Create a new UWP app in C++ Using Foxit PDF SDK for UWP
- Initialize Foxit PDF SDK
- Display a PDF document Using PDFViewCtrl
- Build a full-featured PDF Reader Using UI Extensions Project

3.1.1 Create a new UWP app in C++ Using Foxit PDF SDK for UWP

In this guide, we will create a new UWP project in Visual Studio 2017, and use local machine with x86 as an example to run the project.

1) Create a new project in Visual Studio 2017.

   Open Visual Studio 2017, choose **File** -> **New** -> **Project**... to start the **New Project** wizard. Create a new Windows Universal project called "PDFReader". Please make sure to choose Visual C++ and a blank app (See Figure 3-1).


New Project
Recent
Installed
Visual C#
Visual Basic
Visual C++
Windows Desktop
Cross Platform
MFC/ATL
Windows Universal
Test
Other
Extensibility
JavaScript
Other Project Types
Online
Not finding what you are looking for?
Open Visual Studio Installer
Sort by: Default
Search (Ctrl+E)
Blank App (Universal Windows) Visual C++
DirectX 11 App (Universal Windows) Visual C++
DirectX 12 App (Universal Windows) Visual C++
DirectX 11 and XAML App (Universal Windows) Visual C++
Unit Test App (Universal Windows) Visual C++
DLL (Universal Windows) Visual C++
Static Library (Universal Windows) Visual C++
Windows Runtime Component (Universal Windows) Visual C++
Windows Application Packaging Project Visual C++
Type: Visual C++
A project for a single-page Universal Windows Platform (UWP) app that has no predefined controls or layout.
Browse...
OK
Cancel

Name: PDFReader
Location: C:\uwp_sample\
Solution name: PDFReader
[✓] Create directory for solution
[ ] Create new Git repository

Figure 3-1

2) Add **FoxitRDK** extension to References.

In order to use **FoxitRDK** extension in the project, you must first add a reference to it. In the **Solution Explorer**, right-click the **References** node and select **Add Reference…** as shown in Figure 3-2.

***Note**: If you have not installed **FoxitRDK.vsix** extension, please refer to "Install Foxit PDF SDK extension" to install it at first.*

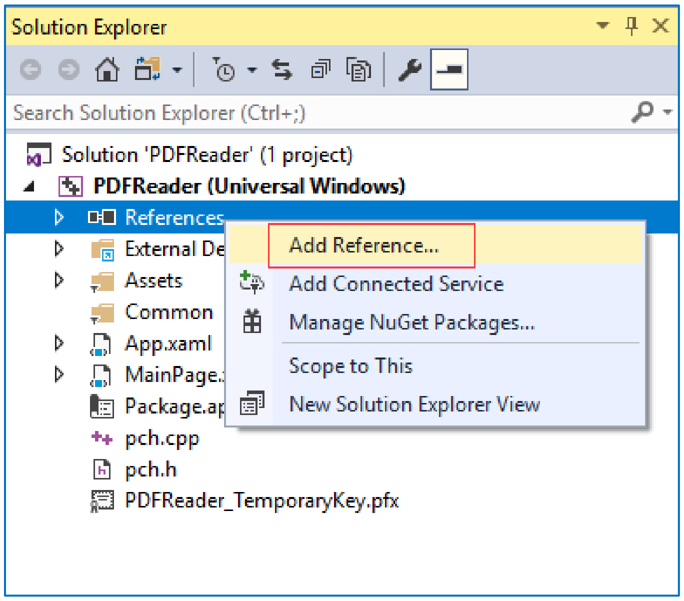
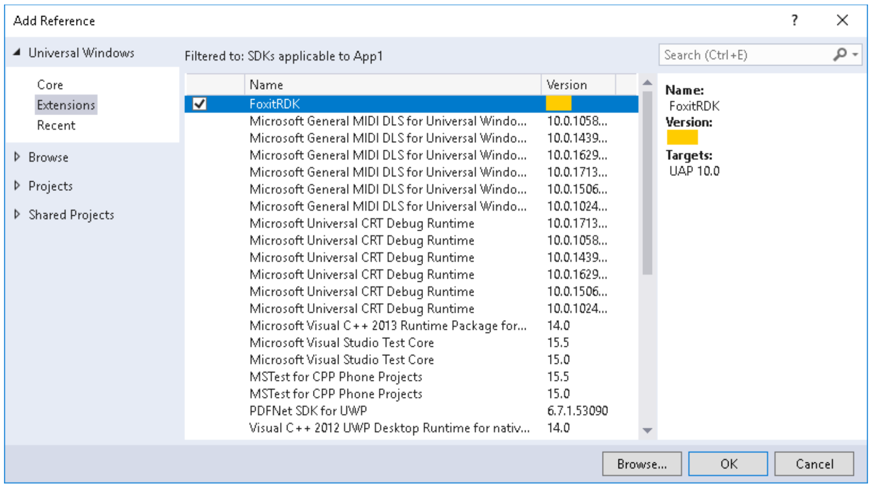


Figure 3-2

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



Filtered to: SDKs applicable to App1

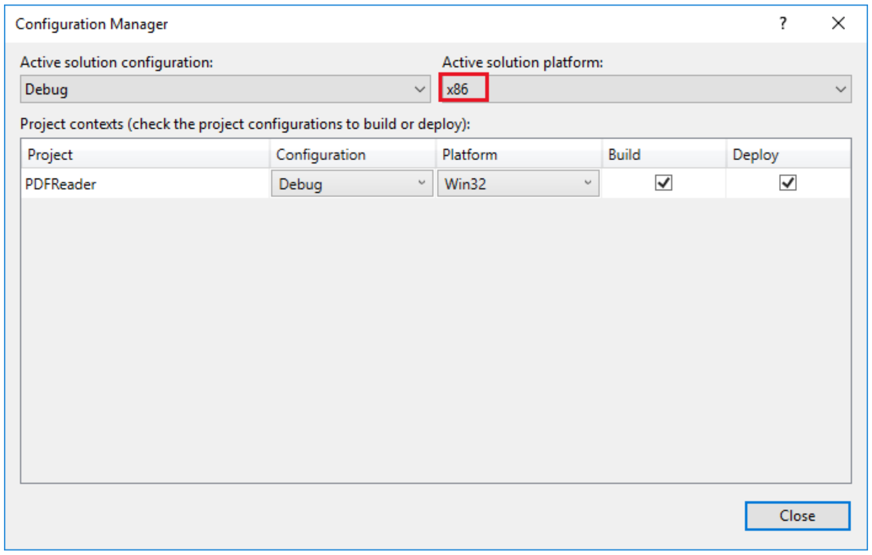
	Name	Version
✓	FoxitRDK	🟨
	Microsoft General MIDI DLS for Universal Windo...	10.0.1058...
	Microsoft General MIDI DLS for Universal Windo...	10.0.1439...
	Microsoft General MIDI DLS for Universal Windo...	10.0.1629...
	Microsoft General MIDI DLS for Universal Windo...	10.0.1713...
	Microsoft General MIDI DLS for Universal Windo...	10.0.1506...
	Microsoft General MIDI DLS for Universal Windo...	10.0.1024...
	Microsoft Universal CRT Debug Runtime	10.0.1713...
	Microsoft Universal CRT Debug Runtime	10.0.1058...
	Microsoft Universal CRT Debug Runtime	10.0.1439...
	Microsoft Universal CRT Debug Runtime	10.0.1629...
	Microsoft Universal CRT Debug Runtime	10.0.1506...
	Microsoft Universal CRT Debug Runtime	10.0.1024...
	Microsoft Visual C++ 2013 Runtime Package for...	14.0
	Microsoft Visual Studio Test Core	15.5
	Microsoft Visual Studio Test Core	15.0
	MSTest for CPP Phone Projects	15.5
	MSTest for CPP Phone Projects	15.0
	PDFNet SDK for UWP	6.7.1.53090
	Visual C++ 2012 UWP Desktop Runtime for nativ...	14.0

Name:
FoxitRDK
Version:
🟨
Targets:
UAP 10.0

Figure 3-3

3) Change the build architecture of the project.

Click on **Build** -> **Configuration Manager** and select **x86** for "Active solution platform" as shown in Figure 3-4.



Active solution configuration: Debug

Active solution platform: x86

Project contexts (check the project configurations to build or deploy):

Project	Configuration	Platform	Build	Deploy
PDFReader	Debug	Win32	✓	✓

Figure 3-4

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 "Active solution platform".*

3.1.2 Initialize Foxit PDF SDK

It is necessary for apps to initialize and unlock Foxit PDF SDK using a license before calling any APIs. The function ***foxit::common::Library::Initialize(sn, key)*** is provided to initialize Foxit PDF SDK. The trial license files can be found in the "libs" folder of the download package. After the evaluation period expires, you should purchase an official license to continue using it. Finish the initialization at the end of the **App::OnLaunched** function within the **App.xaml.cpp** file.

```
// Initialize Foxit PDF SDK.
String^ sn = " ";
String^ key = " ";
foxit::common::ErrorCode ret = foxit::common::Library::Initialize(sn, key);
if (ret != foxit::common::ErrorCode::e_ErrSuccess)
  return;
```

Note *The value of "sn" can be found in the "**rdk_sn.txt**" (the string after "SN=") and the value of "key" can be found in the "**rdk_key.txt**" (the string after "Sign=").*

3.1.3 Display a PDF document Using PDFViewCtrl

So far, we have added **FoxitRDK** extension as a reference and finished the initialization of the Foxit PDF SDK. Now, let's start displaying a PDF document using PDFViewCtrl with just a few lines of code.

1) Add a PDFViewCtrl to display a PDF document.

   Open up "MainPage.xaml", update the whole file as follows:

```
<Page
    x:Class="PDFReader.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:PDFReader"
    xmlns:rdkcontrol="using:foxit.rdk"
    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}">
        <UserControl x:Name="HostContent" Grid.Row="0">
            <!--Place a PDF View control.-->
            <rdkcontrol:PDFViewCtrl x:Name="PDFViewControl">
                <Grid>
                    <!--Place a button on the left-top of the view control.-->
                    <Button Content="Open File" FontSize="20" HorizontalAlignment="Left"
                            VerticalAlignment="Top" Click="OpenFileButton_Click"/>
                </Grid>
            </rdkcontrol:PDFViewCtrl>
        </UserControl>

        <!--Place a holder for uiextension flyout placement.-->
        <Grid x:Name="FlyoutGird" Grid.Column="1" Grid.RowSpan="2" MaxWidth="300">
        </Grid>
    </Grid>
</Page>
```

   Here, we add a button with a click event "OpenFileButton_Click" to open a PDF document, and add a PDFViewCtrl to display the PDF document.

2) Declare and implement "OpenFileButton_Click" function.

   Open up "MainPage.xaml.h", declare the "OpenFileButton_Click" function as follows:

```
#pragma once

#include "MainPage.g.h"

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

    private:
            void OpenFileButton_Click(Platform::Object^ sender, Windows::UI::Xaml::RoutedEventArgs^ e);

    };
}
```

Open up "MainPage.xaml.cpp", implement the "OpenFileButton_Click" function as follows:

```
void PDFReader::MainPage::OpenFileButton_Click(Platform::Object^ sender,
Windows::UI::Xaml::RoutedEventArgs^ e)
{
    // Select a PDF file from a file selection dialog.
    Windows::Storage::Pickers::FileOpenPicker^ openPicker = ref new
Windows::Storage::Pickers::FileOpenPicker();
    openPicker->ViewMode = Windows::Storage::Pickers::PickerViewMode::Thumbnail;
    openPicker->SuggestedStartLocation =
Windows::Storage::Pickers::PickerLocationId::DocumentsLibrary;
    openPicker->FileTypeFilter->Append(".pdf");

    concurrency::create_task(openPicker->PickSingleFileAsync())
            .then([this](Windows::Storage::StorageFile^ file)
    {
            if (file)
            {
                    PDFViewControl->OpenDocAsync(file, "");
            }
    });
}
```

Inside the "OpenFileButton_Click" function, get a PDF document from the file picker, and call **OpenDocAsync** interface to display the PDF document.

3) Click on **Local Machine** to build and run the project. The screenshot of the project is shown in Figure 3-5.

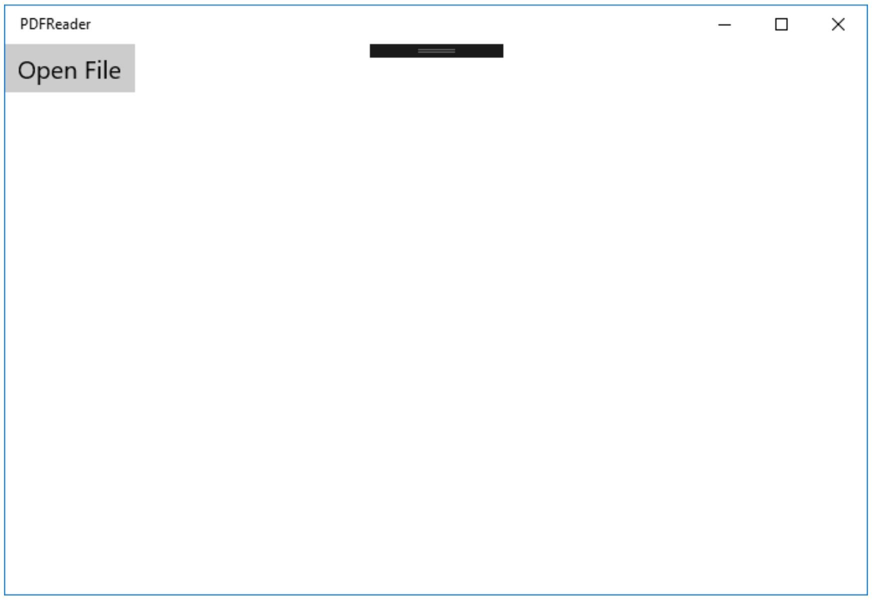


Figure 3-5

Click on **Open File** to choose and open a PDF document. For example, we choose the "sample.pdf" in the "Samples\FunctionDemo\TestFiles" folder, and then it will be displayed as shown in Figure 3-6.

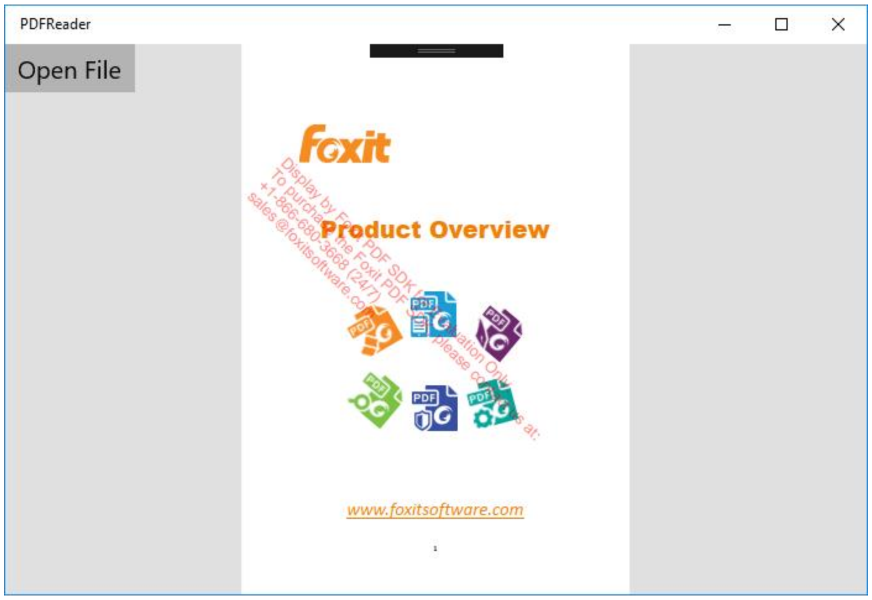
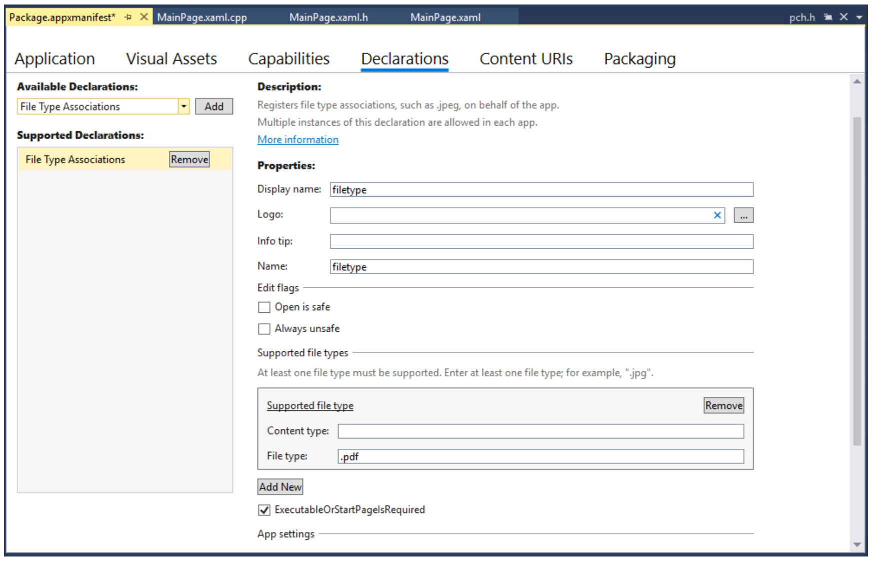


Figure 3-6

Note: *If you cannot open PDF files, you should add File Type Associations Declarations in the "**Package.appxmanifest**" file.*

*Double click **Package.appxmanifest** in the project, find **Declarations**, and add a File Type Associations as shown in Figure 3-7.*



Available Declarations: File Type Associations [Add]

Supported Declarations: File Type Associations [Remove]

Description:
Registers file type associations, such as .jpeg, on behalf of the app.
Multiple instances of this declaration are allowed in each app.
More information

Properties:

Display name: filetype
Logo:
Info tip:
Name: filetype

Edit flags
- [ ] Open is safe
- [ ] Always unsafe

Supported file types
At least one file type must be supported. Enter at least one file type; for example, ".jpg".

Supported file type [Remove]
Content type:
File type: .pdf

[Add New]

- [x] ExecutableOrStartPageIsRequired

App settings

Figure 3-7

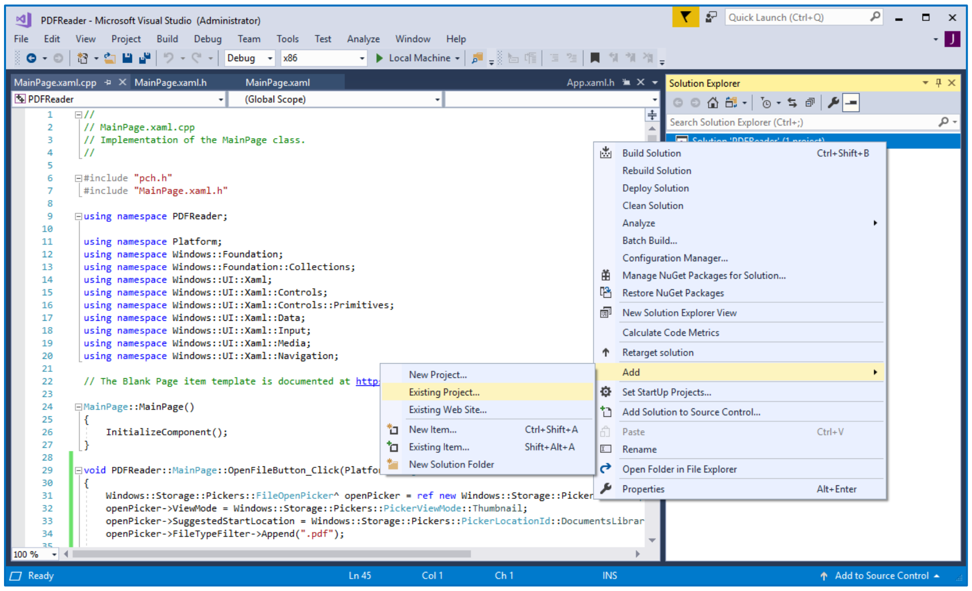
Now, we have finished displaying a PDF document using PDFViewCtrl. At that time, the PDFReader app is just a very basic PDF document viewer. You can only view or zoom in/out the PDF document. If you want to build a full-featured PDF Reader app, just follow the steps in the next section.

3.1.4 Build a full-featured PDF Reader Using UI Extensions Project

The UI Extensions project contains all of the UI implementations including the basic UI for app and the feature modules UI. Hence, building a full-featured PDF Reader is getting simpler and easier. Let's try it just now.

1) Add "**UIExtensions**" project in the "libs" folder of the download package to the solution.

   In the solution explorer, right-click on the **PDFReader** solution and select **Add** -> **Existing Project...**, navigate to the folder where the "foxitpdfsdk_7_0_2_uwp.zip" was unzipped, open the "UIExtensions" folder in the "libs" folder, then select "UIExtensions.vcxproj". (See Figure 3-8 and Figure 3-9)


PDFReader - Microsoft Visual Studio (Administrator)
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Debug x86 Local Machine
MainPage.xaml.cpp MainPage.xaml.h MainPage.xaml App.xaml.h
PDFReader (Global Scope)
Solution Explorer
Search Solution Explorer (Ctrl+;)
Build Solution Ctrl+Shift+B
Rebuild Solution
Deploy Solution
Clean Solution
Analyze
Batch Build...
Configuration Manager...
Manage NuGet Packages for Solution...
Restore NuGet Packages
New Solution Explorer View
Calculate Code Metrics
Retarget solution
Add
Set StartUp Projects...
Add Solution to Source Control...
Paste Ctrl+V
Rename
Open Folder in File Explorer
Properties Alt+Enter
New Project...
Existing Project...
Existing Web Site...
New Item... Ctrl+Shift+A
Existing Item... Shift+Alt+A
New Solution Folder
100 %
Ready Ln 45 Col 1 Ch 1 INS Add to Source Control

```
//
// MainPage.xaml.cpp
// Implementation of the MainPage class.
//

#include "pch.h"
#include "MainPage.xaml.h"

using namespace PDFReader;

using namespace Platform;
using namespace Windows::Foundation;
using namespace Windows::Foundation::Collections;
using namespace Windows::UI::Xaml;
using namespace Windows::UI::Xaml::Controls;
using namespace Windows::UI::Xaml::Controls::Primitives;
using namespace Windows::UI::Xaml::Data;
using namespace Windows::UI::Xaml::Input;
using namespace Windows::UI::Xaml::Media;
using namespace Windows::UI::Xaml::Navigation;

// The Blank Page item template is documented at http

MainPage::MainPage()
{
    InitializeComponent();
}

void PDFReader::MainPage::OpenFileButton_Click(Platfo
{
    Windows::Storage::Pickers::FileOpenPicker^ openPicker = ref new Windows::Storage::Picker
    openPicker->ViewMode = Windows::Storage::Pickers::PickerViewMode::Thumbnail;
    openPicker->SuggestedStartLocation = Windows::Storage::Pickers::PickerLocationId::DocumentsLibrar
    openPicker->FileTypeFilter->Append(".pdf");
```

Figure 3-8

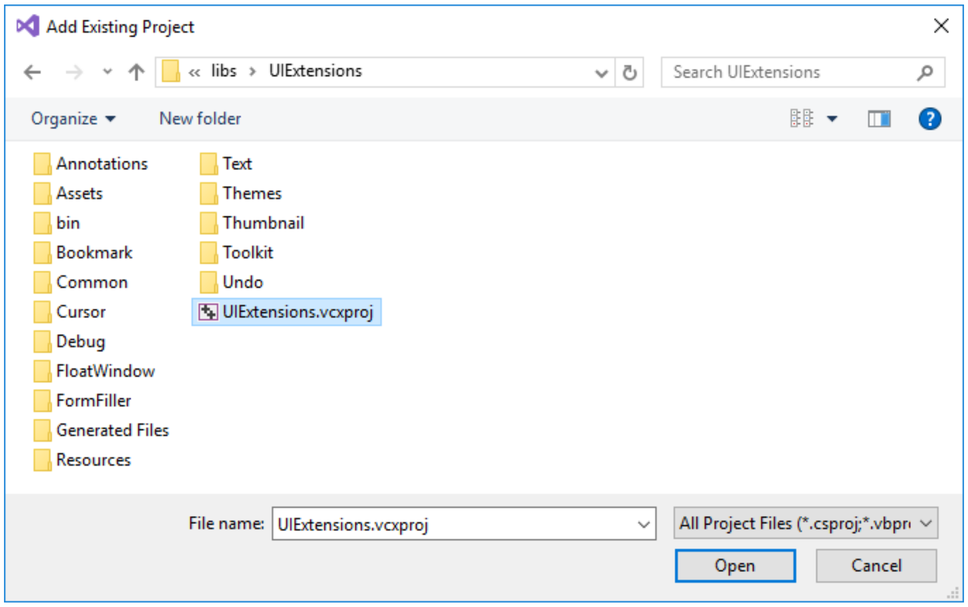
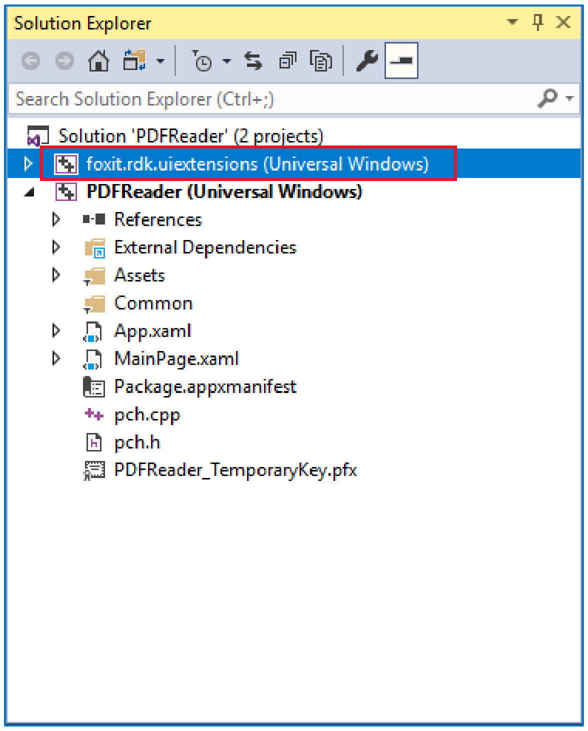


Figure 3-9

After adding the "**UIExtensions**" project, the solution should look like Figure 3-10.

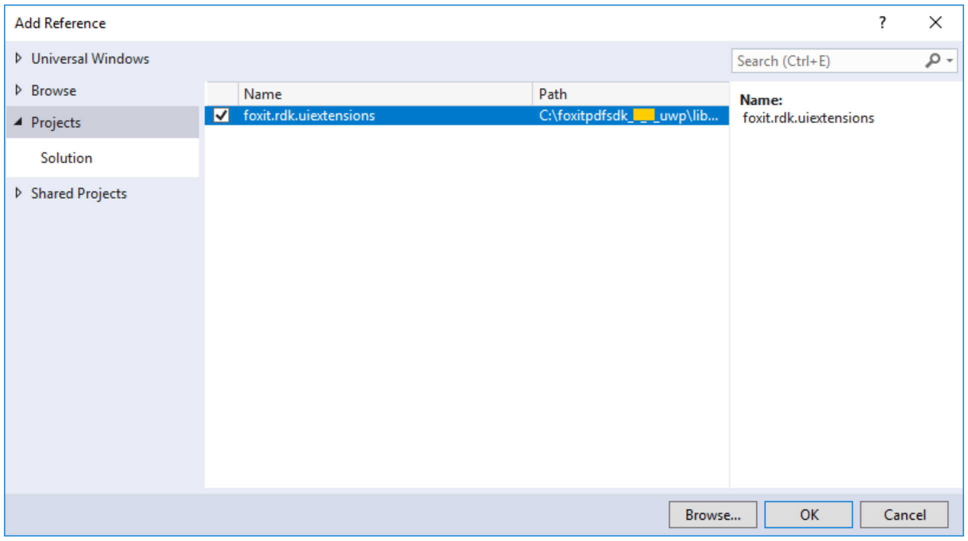


Solution 'PDFReader' (2 projects)
foxit.rdk.uiextensions (Universal Windows)
PDFReader (Universal Windows)
References
External Dependencies
Assets
Common
App.xaml
MainPage.xaml
Package.appxmanifest
pch.cpp
pch.h
PDFReader_TemporaryKey.pfx

Figure 3-10

2) Add "**foxit.rdk.uiextensions**" project as a reference used by PDFReader. Do this in a similar way to how we added **FoxitRDK** extension.

In the **Solution Explorer**, in the **PDFReader** project, right-click the **References** node and select **Add Reference...**. In the Add Reference dialog, select **Projects** -> **Solution** tab, find "foxit.rdk.uiextensions" in the list, check the box next to it and then click **OK**. (See Figure 3-11)



	Name	Path
✓	foxit.rdk.uiextensions	C:\foxitpdfsdk__uwp\lib...

Name:
foxit.rdk.uiextensions

Figure 3-11

3) Build "foxit.rdk.uiextensions" project. Right-click the project, select **Build**.

4) Add SDK resources to the project.

Copy the **Resources** folder from "libs\UIExtensions" folder of Foxit PDF SDK for UWP package to "PDFReader\PDFReader" folder.

In the **Solution Explorer,** click the icon (See Figure 3-12) to show all files. Then, expand the Resources folder, right-click on the items that are marked in red, and choose "**Include In Project**" to include the resource files to the project (For example, see Figure 3-13).

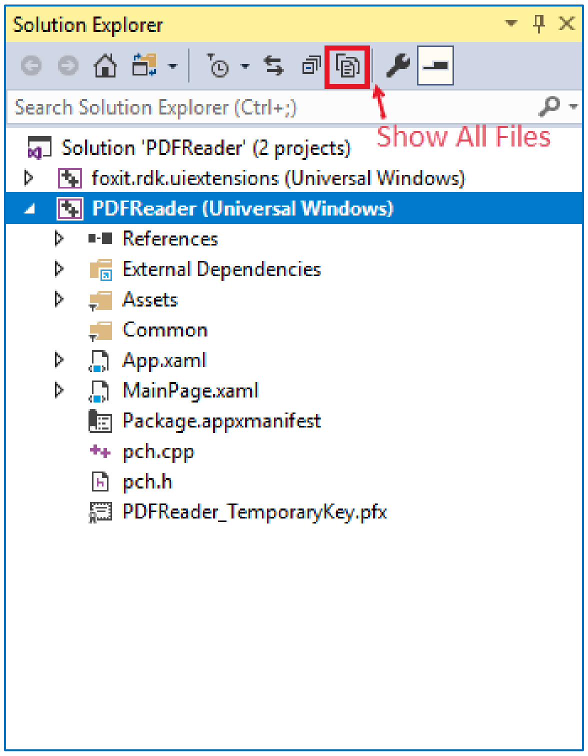

Solution Explorer
Search Solution Explorer (Ctrl+;)
Show All Files
Solution 'PDFReader' (2 projects)
foxit.rdk.uiextensions (Universal Windows)
PDFReader (Universal Windows)
References
External Dependencies
Assets
Common
App.xaml
MainPage.xaml
Package.appxmanifest
pch.cpp
pch.h
PDFReader_TemporaryKey.pfx

Figure 3-12

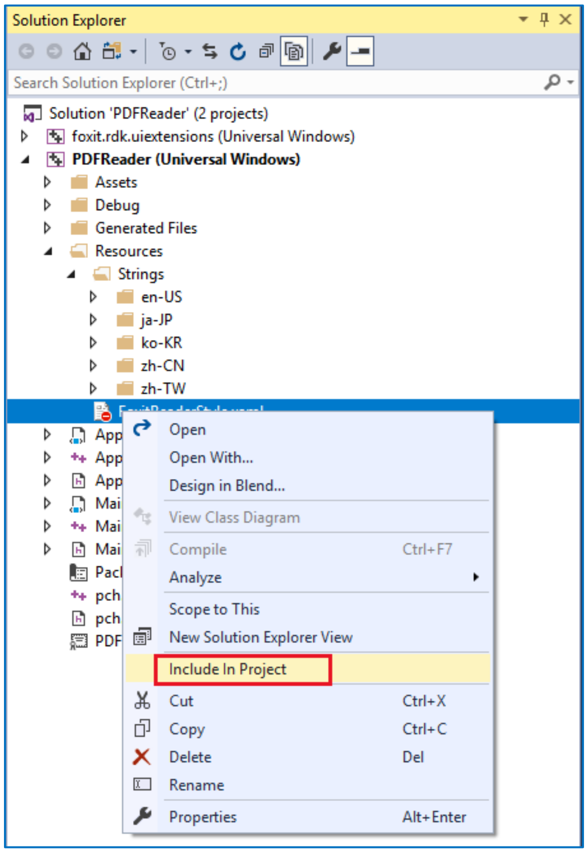
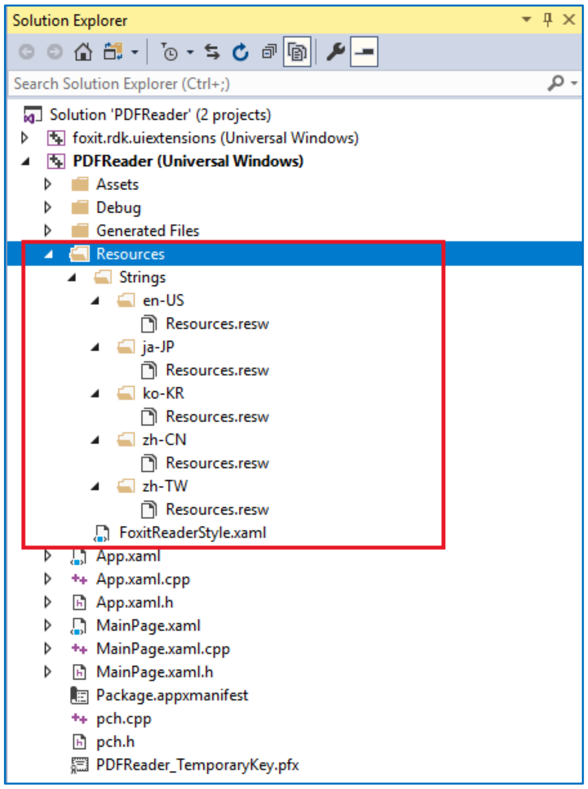


Figure 3-13

After including all the resource items, the PDFReader project should look like Figure 3-14.


Solution Explorer

Search Solution Explorer (Ctrl+;)

```
Solution 'PDFReader' (2 projects)
  foxit.rdk.uiextensions (Universal Windows)
  PDFReader (Universal Windows)
    Assets
    Debug
    Generated Files
    Resources
      Strings
        en-US
          Resources.resw
        ja-JP
          Resources.resw
        ko-KR
          Resources.resw
        zh-CN
          Resources.resw
        zh-TW
          Resources.resw
      FoxitReaderStyle.xaml
    App.xaml
    App.xaml.cpp
    App.xaml.h
    MainPage.xaml
    MainPage.xaml.cpp
    MainPage.xaml.h
    Package.appxmanifest
    pch.cpp
    pch.h
    PDFReader_TemporaryKey.pfx
```

Figure 3-14

5) Instantiate a UIExtensionsManager object and set it to PDFViewCtrl.

Open up "MainPage.xaml.h", declare a UIExtensionsManager object as follows:

```
private:
    foxit::rdk::uiextensions::UIExtensionsManager^ m_uiextensions;
```

Open up "MainPage.xaml.cpp", instantiate the UIExtensionsManager object and set it to PDFViewCtrl in the constructor of MainPage:

```
MainPage::MainPage()
{
    InitializeComponent();

    m_uiextensions = ref new foxit::rdk::uiextensions::UIExtensionsManager(PDFViewControl);

    // Associate the flyout placement of uiextensions with MainPage layout.
    m_uiextensions->SetFlyoutPlacement(FlyoutGird);

    // Set the uiextensions instance to PDF view control.
    PDFViewControl->ExtensionsManager = m_uiextensions;
}
```

6) Click on **Local Machine** to build and run the project.

Click on **Open File** to choose and open a PDF document. For example, we choose the "sample.pdf" in the "Samples\FunctionDemo\TestFiles" folder, and then it will be displayed as shown in Figure 3-15.

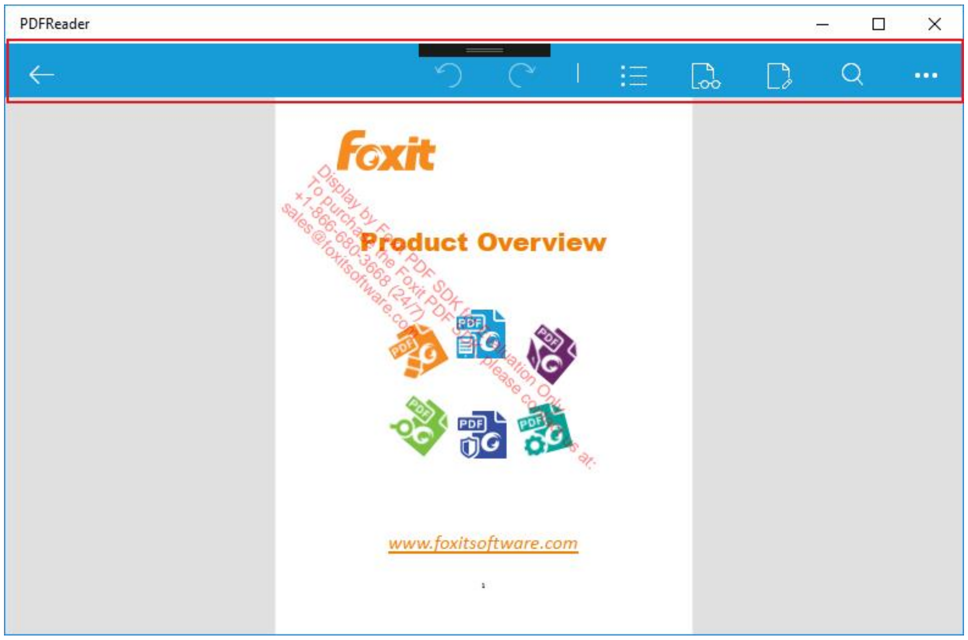


Figure 3-15

Now, a full-featured PDF Reader app which is similar to the Complete PDF viewer demo has been built. Feel free to try the features.

The "**foxit.rdk.uiextensions**" project has implemented the basic UI for app and the feature module UI, so that if you want to customize the UI for styling the appearance as desired, you can change the code in the UIExtensions project directly.

Complete Program

This section shows the whole update of "MainPage.xaml.h" and "MainPage.xaml.cpp".

MainPage.xaml.h

```
//
// MainPage.xaml.h
// Declaration of the MainPage class.
//

#pragma once
```

```
#include "MainPage.g.h"

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

	private:
		void OpenFileButton_Click(Platform::Object^ sender, Windows::UI::Xaml::RoutedEventArgs^ e);

	private:
		foxit::rdk::uiextensions::UIExtensionsManager^ m_uiextensions;

	};
}
```

MainPage.xaml.cpp

```
//
// MainPage.xaml.cpp
// Implementation of the MainPage class.
//

#include "pch.h"
#include "MainPage.xaml.h"

using namespace PDFReader;

using namespace Platform;
using namespace Windows::Foundation;
using namespace Windows::Foundation::Collections;
using namespace Windows::UI::Xaml;
using namespace Windows::UI::Xaml::Controls;
using namespace Windows::UI::Xaml::Controls::Primitives;
using namespace Windows::UI::Xaml::Data;
using namespace Windows::UI::Xaml::Input;
using namespace Windows::UI::Xaml::Media;
using namespace Windows::UI::Xaml::Navigation;

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

MainPage::MainPage()
{
	InitializeComponent();
```

```
        m_uiextensions = ref new foxit::rdk::uiextensions::UIExtensionsManager(PDFViewControl);

        // Associate the flyout placement of uiextensions with MainPage layout.
        m_uiextensions->SetFlyoutPlacement(FlyoutGird);

        // Set the uiextensions instance to PDF view control.
        PDFViewControl->ExtensionsManager = m_uiextensions;
}

void PDFReader::MainPage::OpenFileButton_Click(Platform::Object^ sender,
Windows::UI::Xaml::RoutedEventArgs^ e)
{
        // Select a PDF file from a file selection dialog.
        Windows::Storage::Pickers::FileOpenPicker^ openPicker = ref new
Windows::Storage::Pickers::FileOpenPicker();
        openPicker->ViewMode = Windows::Storage::Pickers::PickerViewMode::Thumbnail;
        openPicker->SuggestedStartLocation =
Windows::Storage::Pickers::PickerLocationId::DocumentsLibrary;
        openPicker->FileTypeFilter->Append(".pdf");

        concurrency::create_task(openPicker->PickSingleFileAsync())
                .then([this](Windows::Storage::StorageFile^ file)
        {
                if (file)
                {
                        PDFViewControl->OpenDocAsync(file, "");
                }
        });
}
```

3.2 Make a UWP app in C# with Foxit PDF SDK for UWP

This section will help you to quickly make a UWP app in C# using Foxit PDF SDK. It includes the following steps:

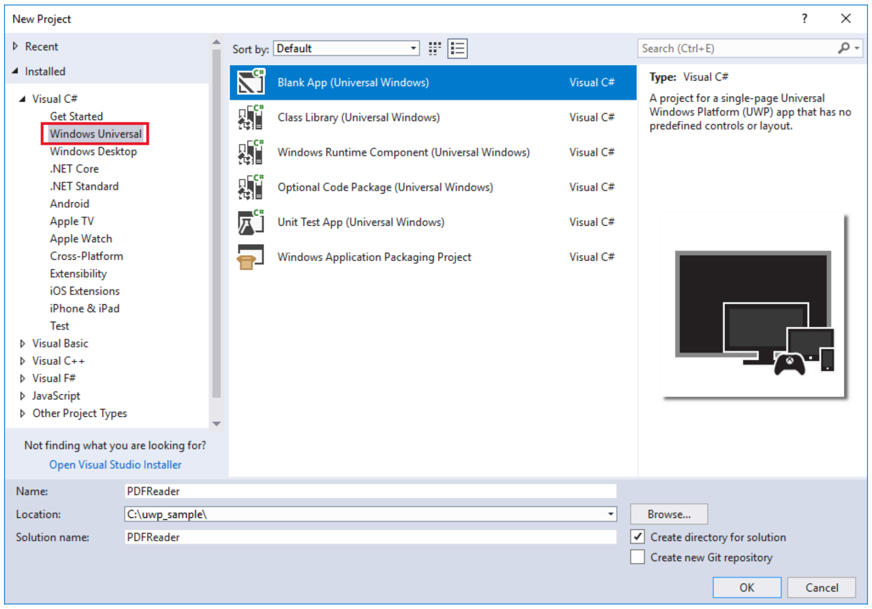
- Create a new UWP app in C# Using Foxit PDF SDK for UWP
- Initialize Foxit PDF SDK
- Display a PDF document Using PDFViewCtrl
- Build a full-featured PDF Reader Using UI Extensions Project

3.2.1 Create a new UWP app in C# Using Foxit PDF SDK for UWP

In this guide, we will create a new UWP project in Visual Studio 2017, and use local machine with x86 as an example to run the project.

1) Create a new project in Visual Studio 2017.

Open Visual Studio 2017, choose **File** -> **New** -> **Project**... to start the **New Project** wizard. Create a new Windows Universal project called "PDFReader". Please make sure to choose Visual C# and a blank app (See Figure 3-16).


New Project
Recent
Installed
Visual C#
Get Started
Windows Universal
Windows Desktop
.NET Core
.NET Standard
Android
Apple TV
Apple Watch
Cross-Platform
Extensibility
iOS Extensions
iPhone & iPad
Test
Visual Basic
Visual C++
Visual F#
JavaScript
Other Project Types
Not finding what you are looking for?
Open Visual Studio Installer
Sort by: Default
Search (Ctrl+E)
Blank App (Universal Windows) Visual C#
Class Library (Universal Windows) Visual C#
Windows Runtime Component (Universal Windows) Visual C#
Optional Code Package (Universal Windows) Visual C#
Unit Test App (Universal Windows) Visual C#
Windows Application Packaging Project Visual C#
Browse...
OK
Cancel

Type: Visual C#

A project for a single-page Universal Windows Platform (UWP) app that has no predefined controls or layout.

Name: PDFReader
Location: C:\uwp_sample\
Solution name: PDFReader
[✓] Create directory for solution
[ ] Create new Git repository

Figure 3-16

2) Add **FoxitRDK** extension to References.

In order to use **FoxitRDK** extension in the project, you must first add a reference to it. In the **Solution Explorer**, right-click the **References** node and select **Add Reference...** as shown in Figure 3-17.

***Note**: If you have not installed **FoxitRDK.vsix** extension, please refer to "Install Foxit PDF SDK extension" to install it at first.*

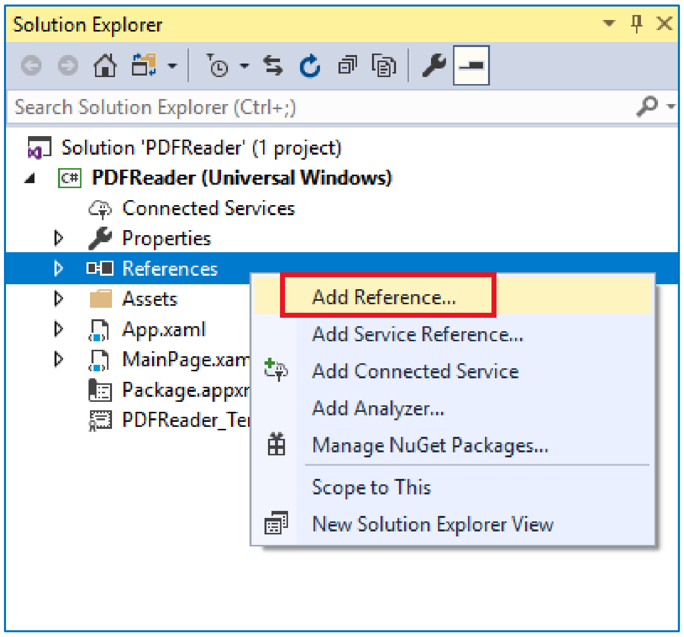
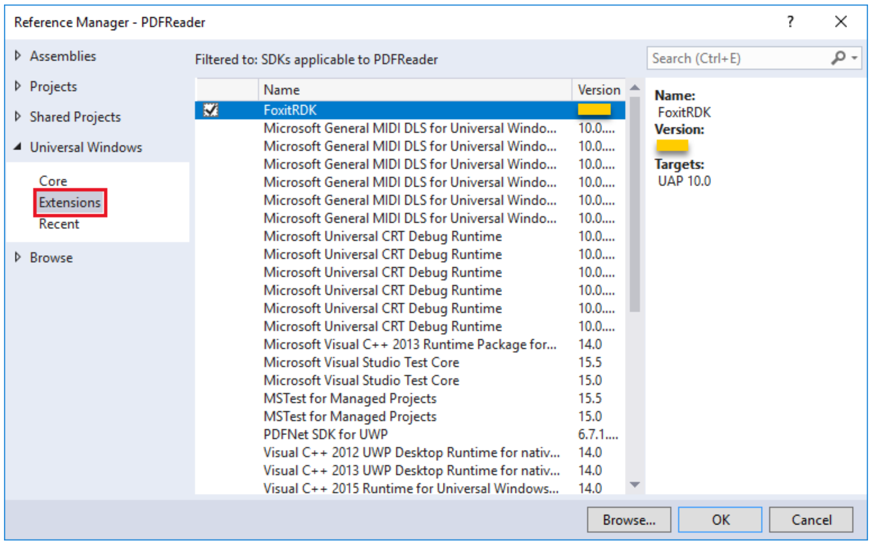


Figure 3-17

In the Reference Manager dialog, select **Universal Windows** -> **Extensions** tab, check the box next to "FoxitRDK", and then click **OK**. It is shown in Figure 3-18.



Filtered to: SDKs applicable to PDFReader

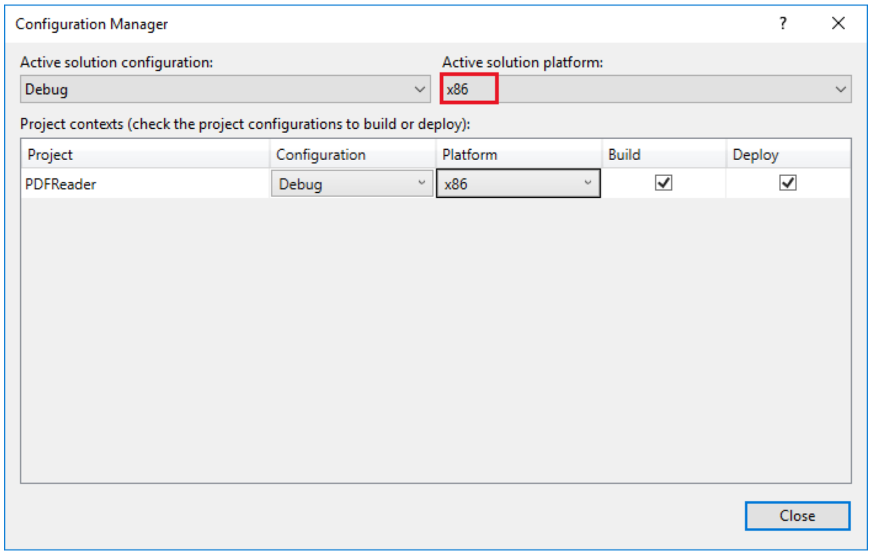
	Name	Version
✓	FoxitRDK	[redacted]
	Microsoft General MIDI DLS for Universal Windo...	10.0....
	Microsoft General MIDI DLS for Universal Windo...	10.0....
	Microsoft General MIDI DLS for Universal Windo...	10.0....
	Microsoft General MIDI DLS for Universal Windo...	10.0....
	Microsoft General MIDI DLS for Universal Windo...	10.0....
	Microsoft General MIDI DLS for Universal Windo...	10.0....
	Microsoft Universal CRT Debug Runtime	10.0....
	Microsoft Universal CRT Debug Runtime	10.0....
	Microsoft Universal CRT Debug Runtime	10.0....
	Microsoft Universal CRT Debug Runtime	10.0....
	Microsoft Universal CRT Debug Runtime	10.0....
	Microsoft Universal CRT Debug Runtime	10.0....
	Microsoft Visual C++ 2013 Runtime Package for...	14.0
	Microsoft Visual Studio Test Core	15.5
	Microsoft Visual Studio Test Core	15.0
	MSTest for Managed Projects	15.5
	MSTest for Managed Projects	15.0
	PDFNet SDK for UWP	6.7.1....
	Visual C++ 2012 UWP Desktop Runtime for nativ...	14.0
	Visual C++ 2013 UWP Desktop Runtime for nativ...	14.0
	Visual C++ 2015 Runtime for Universal Windows...	14.0

Name: FoxitRDK
Version: [redacted]
Targets: UAP 10.0

Figure 3-18

3) Change the build architecture of the project.

Click on **Build** -> **Configuration Manager** and select **x86** for "Active solution platform" as shown in Figure 3-19.



Active solution configuration: Debug

Active solution platform: x86

Project contexts (check the project configurations to build or deploy):

Project	Configuration	Platform	Build	Deploy
PDFReader	Debug	x86	✓	✓

Figure 3-19

***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 "Active solution platform".*

3.2.2 Initialize Foxit PDF SDK

It is necessary for apps to initialize and unlock Foxit PDF SDK using a license before calling any APIs. The function ***foxit.common.Library.Initialize(sn, key)*** is provided to initialize Foxit PDF SDK. The trial license files can be found in the "libs" folder of the download package. After the evaluation period expires, you should purchase an official license to continue using it. Finish the initialization at the end of the **OnLaunched** function within the **App.xaml.cs** file.

```
// Initialize Foxit PDF SDK.
string sn = " ";
string key = " ";
foxit.common.ErrorCode ret = foxit.common.Library.Initialize(sn, key);
if (ret != foxit.common.ErrorCode.e_ErrSuccess)
  return;
```

***Note** The value of "sn" can be found in the "**rdk_sn.txt**" (the string after "SN=") and the value of "key" can be found in the "**rdk_key.txt**" (the string after "Sign=").*

3.2.3 Display a PDF document Using PDFViewCtrl

So far, we have added **FoxitRDK** extension as a reference and finished the initialization of the Foxit PDF SDK. Now, let's start displaying a PDF document using PDFViewCtrl with just a few lines of code.

4) Add a PDFViewCtrl to display a PDF document.

   Open up "MainPage.xaml", update the whole file as follows:

```
<Page
   x:Class="PDFReader.MainPage"
   xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
   xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
   xmlns:local="using:PDFReader"
   xmlns:rdkcontrol="using:foxit.rdk"
   xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
   xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
   mc:Ignorable="d"
   Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

   <Grid>
      <UserControl x:Name="HostContent" Grid.Row="0">
         <!--Place a PDF View control.-->
         <rdkcontrol:PDFViewCtrl x:Name="PDFViewControl">
            <Grid>
               <!--Place a button on the left-top of the view control.-->
               <Button Content="Open File" FontSize="20" HorizontalAlignment="Left"
                     VerticalAlignment="Top" Click="OpenFileButton_Click"/>
            </Grid>
         </rdkcontrol:PDFViewCtrl>
      </UserControl>

      <!--Place a holder for uiextension flyout placement.-->
      <Grid x:Name="FlyoutGird" Grid.Column="1" Grid.RowSpan="2" MaxWidth="300">
      </Grid>
   </Grid>
</Page>
```

   Here, we add a button with a click event "OpenFileButton_Click" to open a PDF document, and add a PDFViewCtrl to display the PDF document.

5) Implement the "OpenFileButton_Click" function.

   Open up "MainPage.xaml.cs", implement the "OpenFileButton_Click" function as follows:

```
private async void OpenFileButton_Click(object sender, RoutedEventArgs e)

{
       // Select a PDF file from a file selection dialog.
```

```
    var fileOpenPicker = new Windows.Storage.Pickers.FileOpenPicker();
    fileOpenPicker.ViewMode = Windows.Storage.Pickers.PickerViewMode.Thumbnail;
    fileOpenPicker.SuggestedStartLocation = Windows.Storage.Pickers.PickerLocationId.DocumentsLibrary;
    fileOpenPicker.FileTypeFilter.Add(".pdf");

    // Open a PDF file asynchronously.
    Windows.Storage.StorageFile file = await fileOpenPicker.PickSingleFileAsync();
    if (file == null)
        return;
    foxit.common.ErrorCode errCode = await this.PDFViewControl.OpenDocAsync(file, "");
}
```

Inside the "OpenFileButton_Click" function, get a PDF document from the file picker, and call **OpenDocAsync** interface to display the PDF document.

6) Click on **Local Machine** to build and run the project. The screenshot of the project is shown in Figure 3-20.

PDFReader
Open File

Figure 3-20

Click on **Open File** to choose and open a PDF document. For example, we choose the "sample.pdf" in the "Samples\FunctionDemo\TestFiles" folder, and then it will be displayed as shown in Figure 3-21.

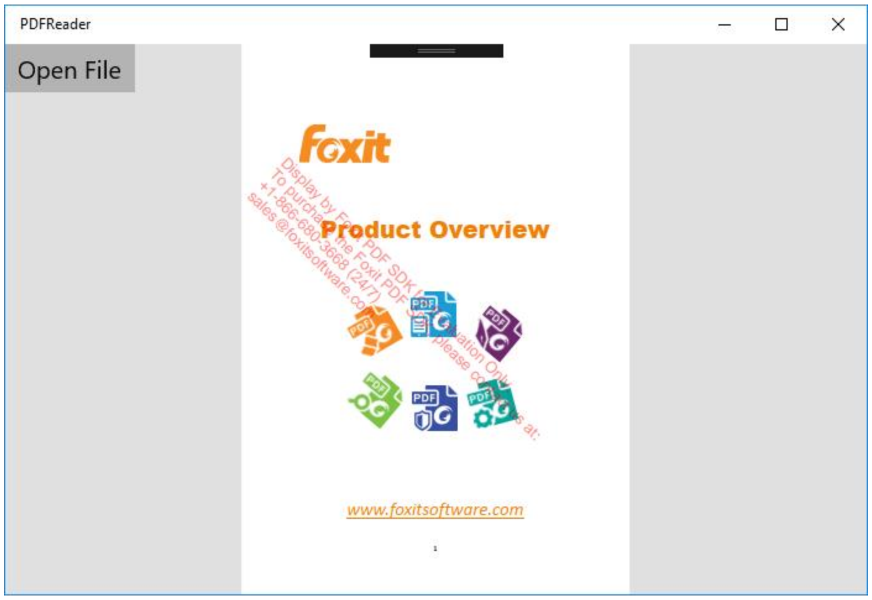
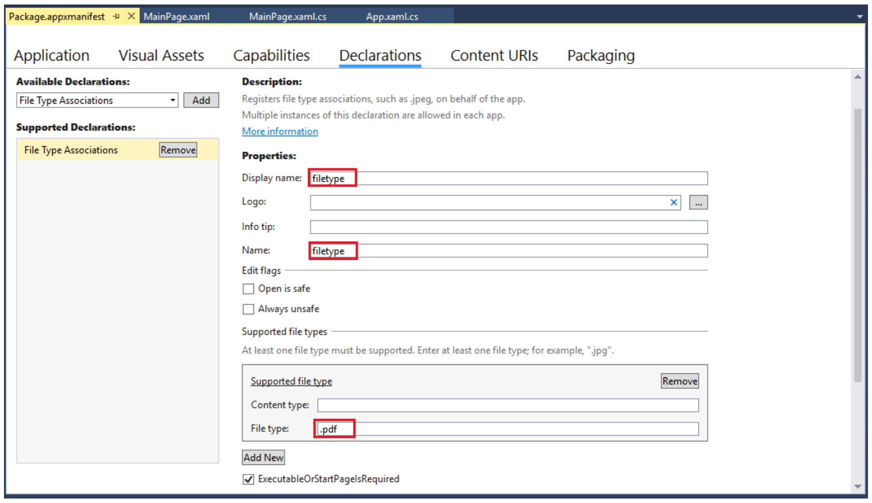


Figure 3-21

Note: *If you cannot open PDF files, you should add File Type Associations Declarations in the "****Package.appxmanifest****" file.*

Double click ***Package.appxmanifest*** *in the project, find* ***Declarations****, and add a File Type Associations as shown in Figure 3-22.*



Available Declarations: File Type Associations

Supported Declarations: File Type Associations

Description:
Registers file type associations, such as .jpeg, on behalf of the app.
Multiple instances of this declaration are allowed in each app.

Properties:
Display name: filetype
Logo:
Info tip:
Name: filetype
Edit flags
☐ Open is safe
☐ Always unsafe
Supported file types
At least one file type must be supported. Enter at least one file type; for example, ".jpg".
Supported file type
Content type:
File type: .pdf
☑ ExecutableOrStartPageIsRequired

Figure 3-22

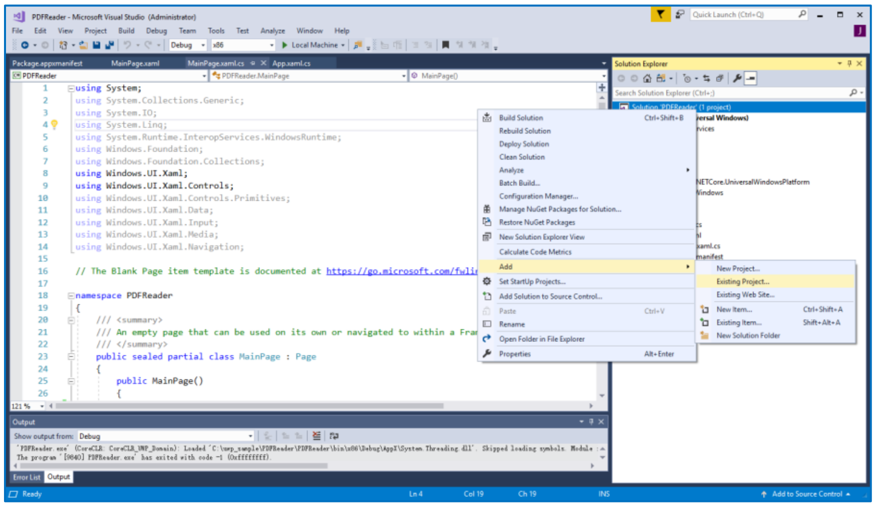
Now, we have finished displaying a PDF document using PDFViewCtrl. At that time, the PDFReader app is just a very basic PDF document viewer. You can only view or zoom in/out the PDF document. If you want to build a full-featured PDF Reader app, just follow the steps in the next section.

3.2.4 Build a full-featured PDF Reader Using UI Extensions Project

The UI Extensions project contains all of the UI implementations including the basic UI for app and the feature modules UI. Hence, building a full-featured PDF Reader is getting simpler and easier. Let's try it just now.

1) Add "**UIExtensions**" project in the "libs" folder of the download package to the solution.

   In the solution explorer, right-click on the **PDFReader** solution and select **Add** -> **Existing Project...**, navigate to the folder where the "foxitpdfsdk_7_0_2_uwp.zip" was unzipped, open the "UIExtensions" folder in the "libs" folder, then select "UIExtensions.vcxproj". (See Figure 3-23 and Figure 3-24)



```
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;

// The Blank Page item template is documented at https://go.microsoft.com/fwlir

namespace PDFReader
{
    /// <summary>
    /// An empty page that can be used on its own or navigated to within a Fran
    /// </summary>
    public sealed partial class MainPage : Page
    {
        public MainPage()
        {
```

```
'PDFReader.exe' (CoreCLR: CoreCLR_UWP_Domain): Loaded 'C:\uwp_sample\PDFReader\PDFReader\bin\x86\Debug\AppX\System.Threading.dll'. Skipped loading symbols. Module :
The program '[9840] PDFReader.exe' has exited with code -1 (0xffffffff).
```

Figure 3-23

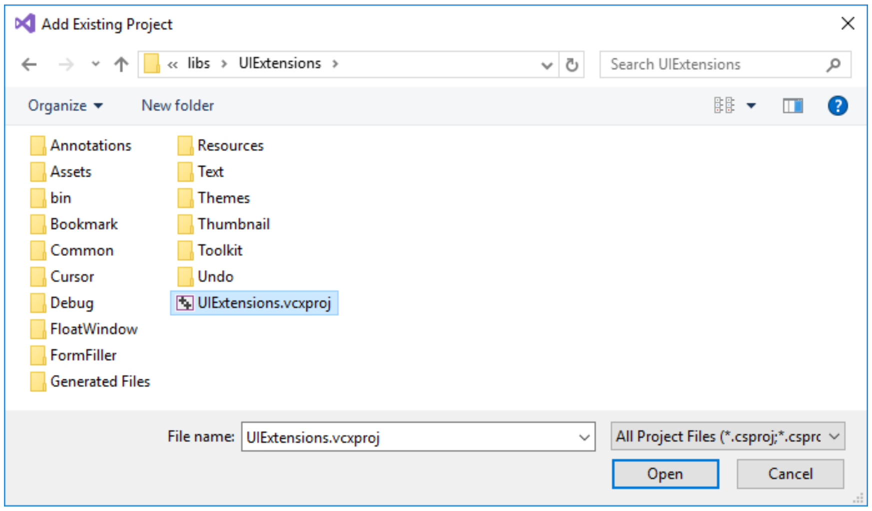


Figure 3-24

After adding the "**UIExtensions**" project, the solution should look like Figure 3-25.

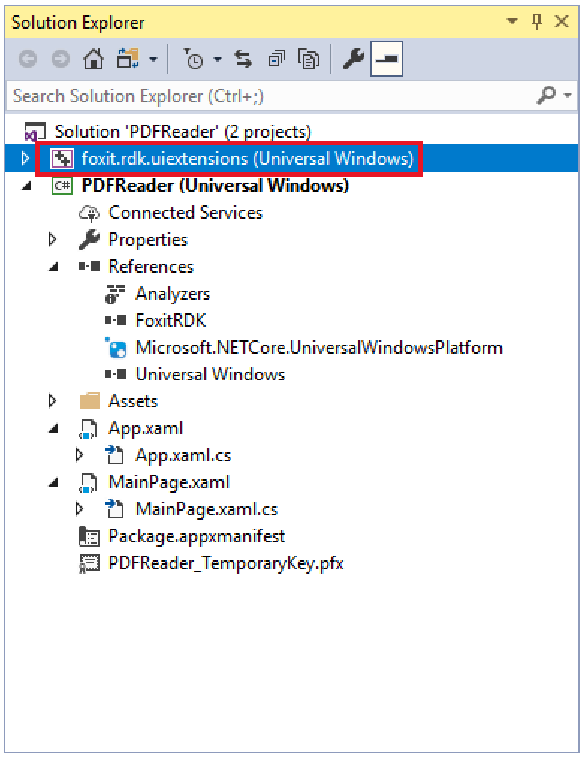

Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'PDFReader' (2 projects)
foxit.rdk.uiextensions (Universal Windows)
PDFReader (Universal Windows)
Connected Services
Properties
References
Analyzers
FoxitRDK
Microsoft.NETCore.UniversalWindowsPlatform
Universal Windows
Assets
App.xaml
App.xaml.cs
MainPage.xaml
MainPage.xaml.cs
Package.appxmanifest
PDFReader_TemporaryKey.pfx

Figure 3-25

2) Add "**foxit.rdk.uiextensions**" project as a reference used by PDFReader. Do this in a similar way to how we added **FoxitRDK** extension.

In the **Solution Explorer**, in the **PDFReader** project, right-click the **References** node and select **Add Reference…**. In the Reference Manager dialog, select **Projects** -> **Solution** tab, find "foxit.rdk.uiextensions" in the list, check the box next to it and then click **OK**. (See Figure 3-26)

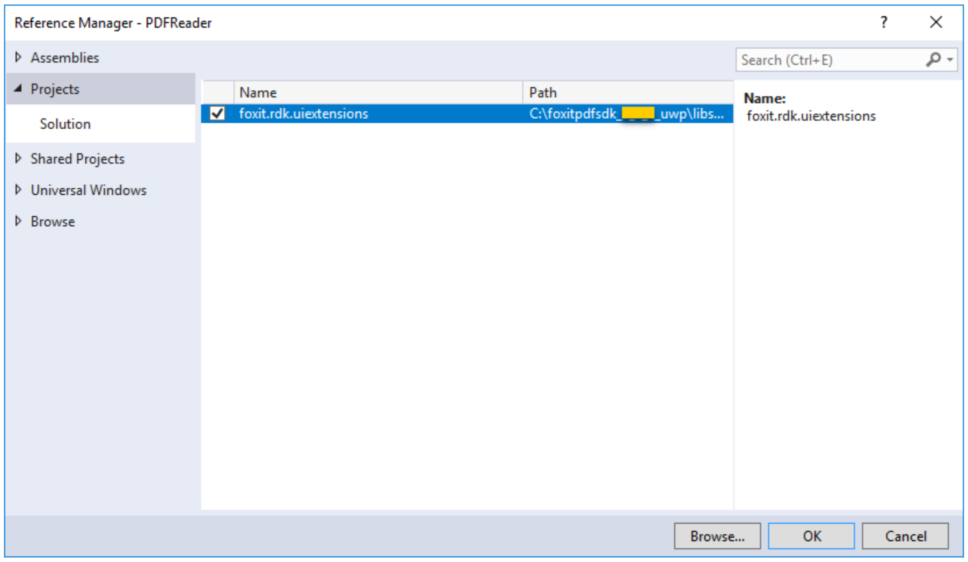


Figure 3-26

3) Build "foxit.rdk.uiextensions" project. Right-click the project, select **Build**.

4) Add SDK resources to the project.

In the **Solution Explorer**, click the **PDFReader** project, find the **Resources** folder under "libs\UIExtensions" folder of Foxit PDF SDK for UWP package, then drag and drop the Resources folder to the PDFReader project. After adding, the PDFReader project should look like Figure 3-27.

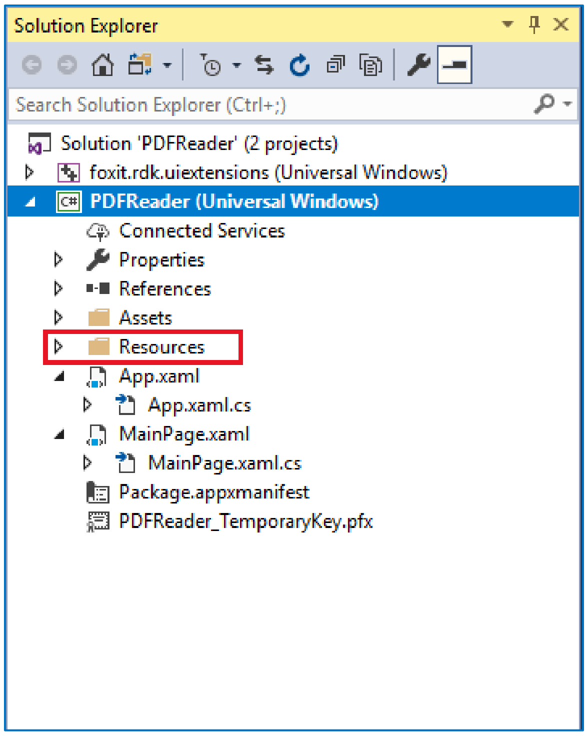

Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'PDFReader' (2 projects)
foxit.rdk.uiextensions (Universal Windows)
PDFReader (Universal Windows)
Connected Services
Properties
References
Assets
Resources
App.xaml
App.xaml.cs
MainPage.xaml
MainPage.xaml.cs
Package.appxmanifest
PDFReader_TemporaryKey.pfx

Figure 3-27

5) Instantiate a UIExtensionsManager object and set it to PDFViewCtrl.

   Open up "MainPage.xaml.cs", instantiate a UIExtensionsManager object and set it to PDFViewCtrl in the constructor of MainPage:

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

    foxit.rdk.uiextensions.UIExtensionsManager m_uiextensions = new
foxit.rdk.uiextensions.UIExtensionsManager(this.PDFViewControl);

    // Associate the flyout placement of uiextensions with MainPage layout.
    m_uiextensions.SetFlyoutPlacement(this.FlyoutGird);

    // Set the uiextensions instance to PDF view control.
    this.PDFViewControl.ExtensionsManager = m_uiextensions;
}
```

6) Click on **Local Machine** to build and run the project.

   Click on **Open File** to choose and open a PDF document. For example, we choose the "sample.pdf" in the "Samples\FunctionDemo\TestFiles" folder, and then it will be displayed as shown in Figure 3-28.

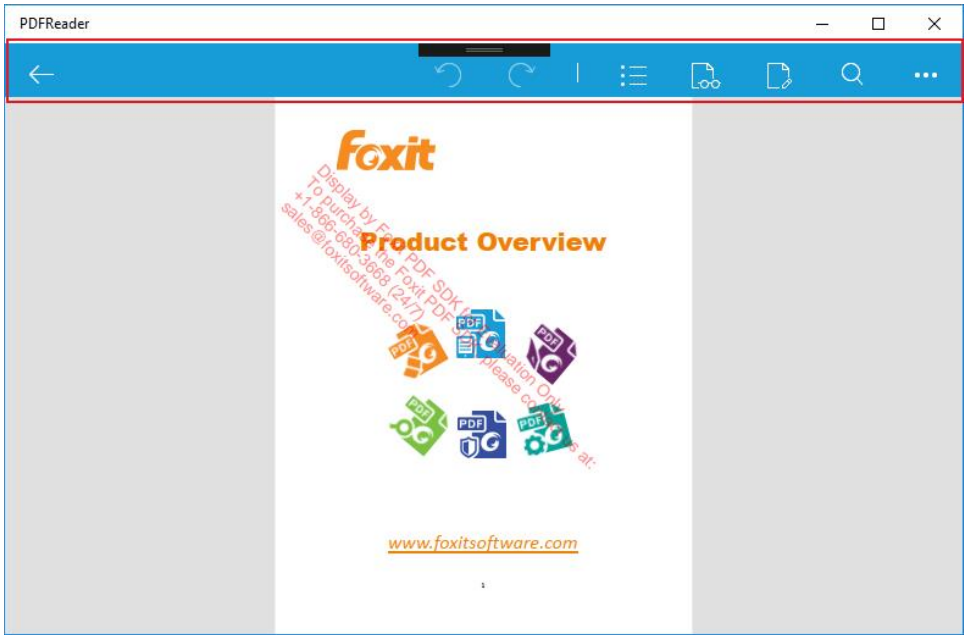

PDFReader
Foxit
Product Overview
Display by Foxit PDF SDK, for evaluation Only
To purchase the Foxit PDF SDK, please contact us at:
+1-866-680-3668 (24/7)
sales@foxitsoftware.com
www.foxitsoftware.com

Figure 3-28

Now, a full-featured PDF Reader app which is similar to the C# demo (PDFViewer) has been built. Feel free to try the features.

The "**foxit.rdk.uiextensions**" project coded by Visual C++ has implemented the basic UI for app and the feature module UI, so that if you want to customize the UI for styling the appearance as desired, you should modify the code in the UIExtensions project directly with Visual C++.

Complete Program

This section shows the whole update of "MainPage.xaml.cs".

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;
```

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

namespace PDFReader
{
    /// <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();

            foxit.rdk.uiextensions.UIExtensionsManager m_uiextensions = new
foxit.rdk.uiextensions.UIExtensionsManager(this.PDFViewControl);

            // Associate the flyout placement of uiextensions with MainPage layout.
            m_uiextensions.SetFlyoutPlacement(this.FlyoutGird);

            // Set the uiextensions instance to PDF view control.
            this.PDFViewControl.ExtensionsManager = m_uiextensions;
        }

        private async void OpenFileButton_Click(object sender, RoutedEventArgs e)
        {
            // Select a PDF file from a file selection dialog.
            var fileOpenPicker = new Windows.Storage.Pickers.FileOpenPicker();
            fileOpenPicker.ViewMode = Windows.Storage.Pickers.PickerViewMode.Thumbnail;
            fileOpenPicker.SuggestedStartLocation = Windows.Storage.Pickers.PickerLocationId.DocumentsLibrary;
            fileOpenPicker.FileTypeFilter.Add(".pdf");

            // Open a PDF file asynchronously.
            Windows.Storage.StorageFile file = await fileOpenPicker.PickSingleFileAsync();
            if (file == null)
                return;
            foxit.common.ErrorCode errCode = await this.PDFViewControl.OpenDocAsync(file, "");
        }
    }
}
```

4 Technical Support

Reporting Problems

Foxit offers 24/7 support for its products and are fully supported by the PDF industry's largest development team of support engineers. If you encounter any technical questions or bug issues when using Foxit PDF SDK for UWP, please submit the problem report to the Foxit support team at http://tickets.foxitsoftware.com/create.php. In order to better help you solve the problem, please provide the following information:

- Contact details
- Foxit PDF SDK product and version
- Your Operating System and IDE version
- Detailed description of the problem
- Any other related information, such as log file or error screenshot

Contact Information

You can contact Foxit directly, please use the contact information as follows:

Foxit Support:

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

Sales Contact:

- 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