Foxit

DEVELOPER GUIDE

Foxit® PDF SDK

For .NET

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

Foxit PDF SDK for .NET 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 into their own applications based on .NET Framework. It aims at introducing installation package structure, 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 .NET 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 into the projects based on .NET Framework. 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 XP, Vista, 7 and 8, 10 (32-bit, 64-bit)

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

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

Visual Studio 2010 installed with .NET Framework 4.0

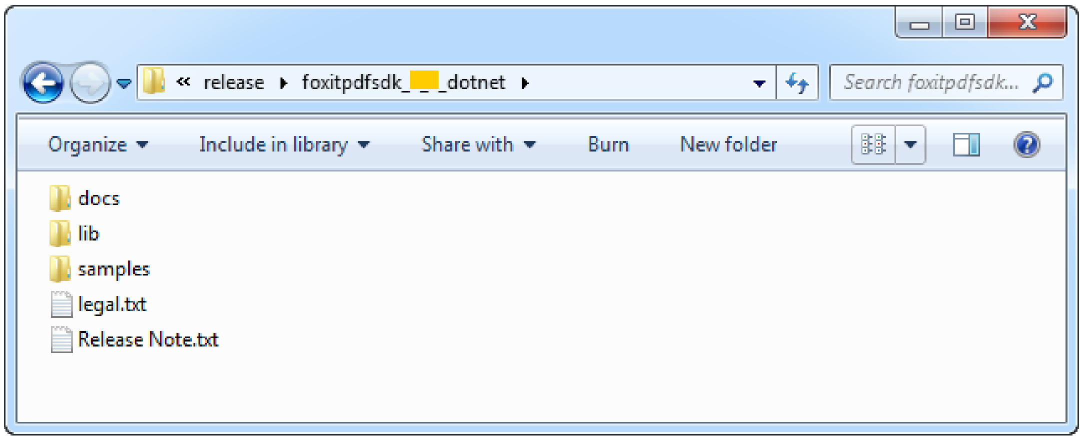
Note:

- It only supports for Windows 8/10 classic style not for Store App.
- .NET Framework 3.5 is not supported from version 5.3.

3.2 What is in the Package

Download Foxit PDF SDK zip for .NET and extract it to a new directory like "foxitpdfsdk_5_3_dotnet". 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:	libraries and license files
samples:	sample project



docs
lib
samples
legal.txt
Release Note.txt

Figure 3-1

3.3 How to run a demo

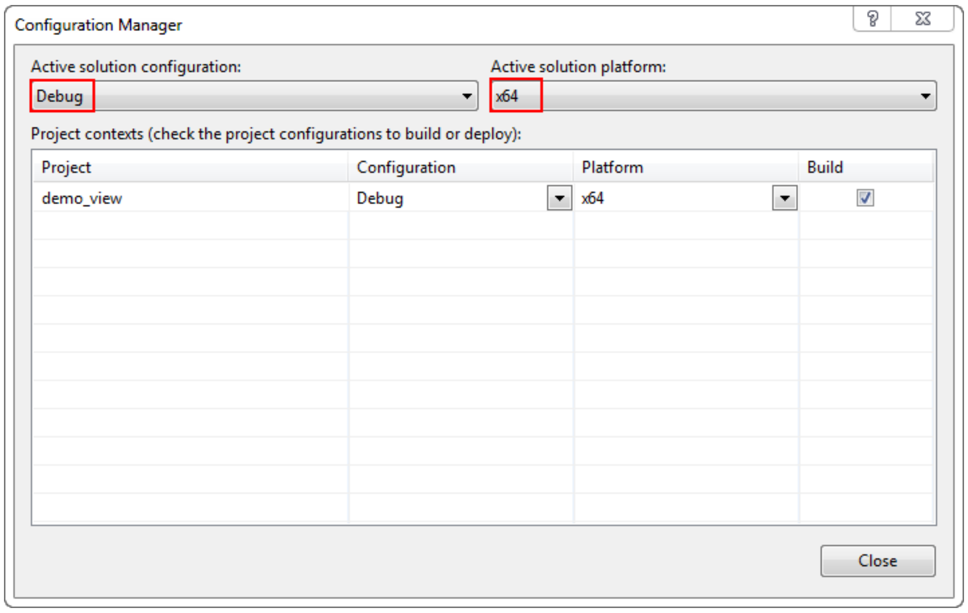
Foxit PDF SDK for .NET provides a simple viewer demo in “samples” folder.

demo_view

demo_view project provides an example for .NET developers on how to implement a simple PDF viewer on Windows platform using Foxit PDF SDK APIs. To run the demo in Visual Studio 2010, follow the steps below:

a) Open the **demo_view.sln** file under “samples\demo_view” in Visual Studio 2010.

b) Choose the build architecture you want for the project. Click on **Build** -> **Configuration Manager…**, and select **Debug** for the “Active solution configuration”, and **x64** for the “Active solution platform” as shown in Figure 3-2. (In this guide, we choose Debug and x64 to build the project.)


Configuration Manager

Active solution configuration: Debug

Active solution platform: x64

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

Project	Configuration	Platform	Build
demo_view	Debug	x64	✓

Figure 3-2

c) Build the project by right clicking the project and selecting **Build**. Then, Press **F5** to run the project, and the screenshot is shown in Figure 3-3.

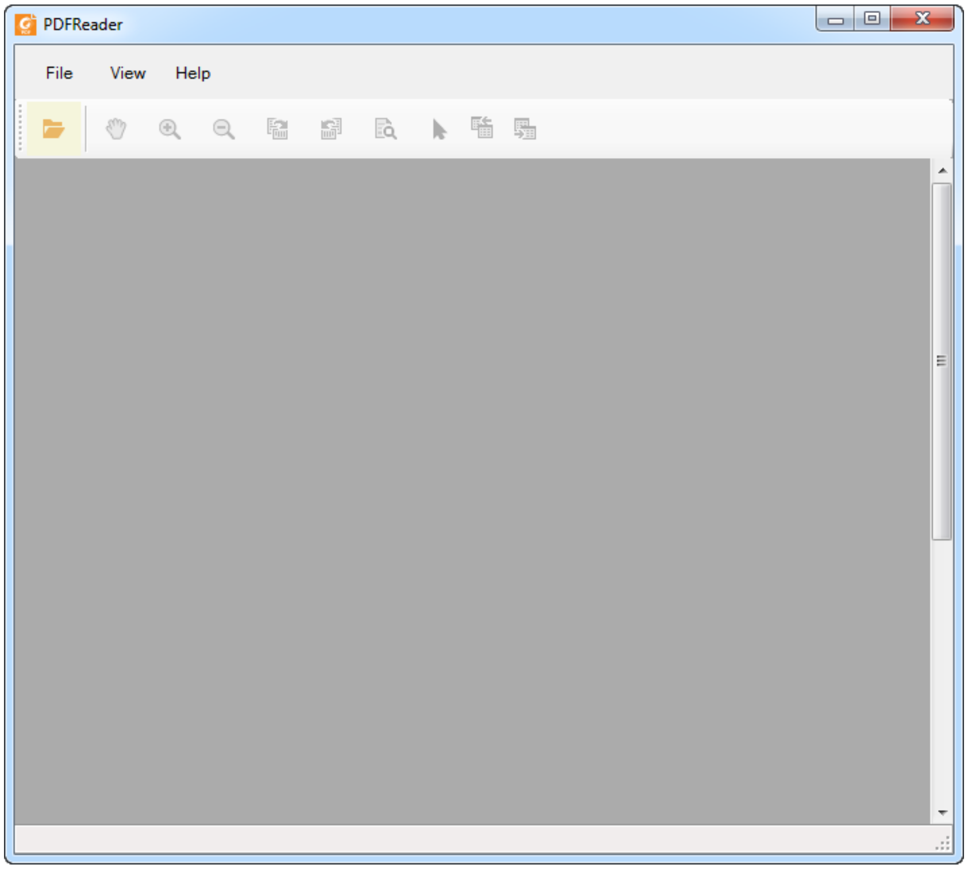
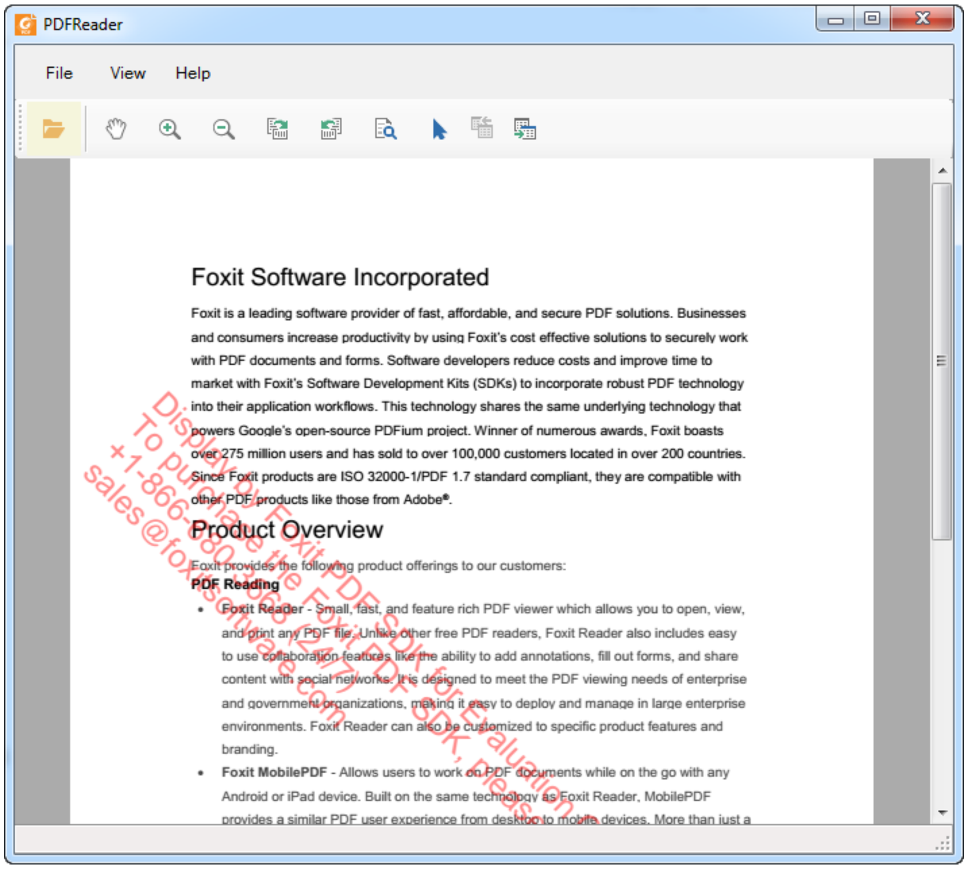


Figure 3-3

d) Click on **File**->**Open** or click the directory icon to open a PDF file. Here, we open a PDF document named “AboutFoxit.pdf”, and it will be displayed as shown in Figure 3-4.



Foxit Software Incorporated

Foxit is a leading software provider of fast, affordable, and secure PDF solutions. Businesses and consumers increase productivity by using Foxit's cost effective solutions to securely work with PDF documents and forms. Software developers reduce costs and improve time to market with Foxit's Software Development Kits (SDKs) to incorporate robust PDF technology into their application workflows. This technology shares the same underlying technology that powers Google's open-source PDFium project. Winner of numerous awards, Foxit boasts over 275 million users and has sold to over 100,000 customers located in over 200 countries. Since Foxit products are ISO 32000-1/PDF 1.7 standard compliant, they are compatible with other PDF products like those from Adobe®.

Product Overview

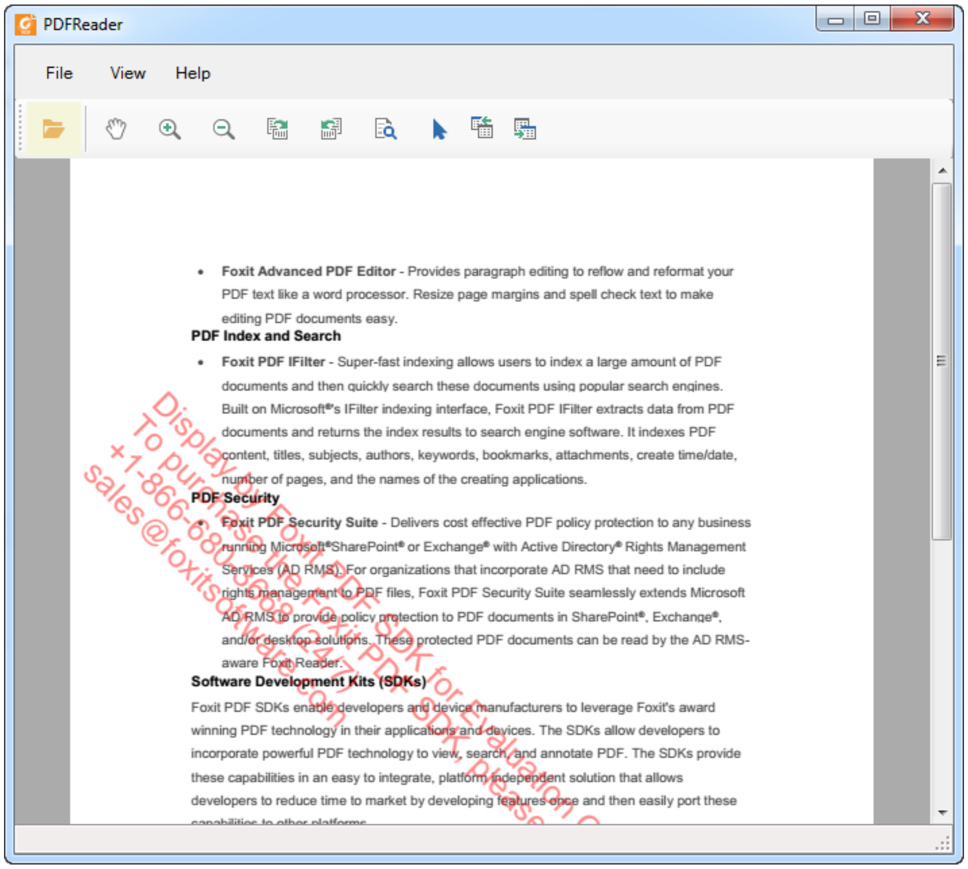
Foxit provides the following product offerings to our customers:

PDF Reading

- **Foxit Reader** - Small, fast, and feature rich PDF viewer which allows you to open, view, and print any PDF file. Unlike other free PDF readers, Foxit Reader also includes easy to use collaboration features like the ability to add annotations, fill out forms, and share content with social networks. It is designed to meet the PDF viewing needs of enterprise and government organizations, making it easy to deploy and manage in large enterprise environments. Foxit Reader can also be customized to specific product features and branding.
- **Foxit MobilePDF** - Allows users to work on PDF documents while on the go with any Android or iPad device. Built on the same technology as Foxit Reader, MobilePDF provides a similar PDF user experience from desktop to mobile devices. More than just a

Figure 3-4

e) This demo provides the features like rendering a PDF document, zooming, page rotation, text selection and search, and page turning. For example, browse the content by scrolling down or moving the PDF page by holding the left mouse button, and then click the Next Page button to view the next page, which is shown in Figure 3-5.



- **Foxit Advanced PDF Editor** - Provides paragraph editing to reflow and reformat your PDF text like a word processor. Resize page margins and spell check text to make editing PDF documents easy.

PDF Index and Search

- **Foxit PDF IFilter** - Super-fast indexing allows users to index a large amount of PDF documents and then quickly search these documents using popular search engines. Built on Microsoft®'s IFilter indexing interface, Foxit PDF IFilter extracts data from PDF documents and returns the index results to search engine software. It indexes PDF content, titles, subjects, authors, keywords, bookmarks, attachments, create time/date, number of pages, and the names of the creating applications.

PDF Security

- **Foxit PDF Security Suite** - Delivers cost effective PDF policy protection to any business running Microsoft® SharePoint® or Exchange® with Active Directory® Rights Management Services (AD RMS). For organizations that incorporate AD RMS that need to include rights management to PDF files, Foxit PDF Security Suite seamlessly extends Microsoft AD RMS to provide policy protection to PDF documents in SharePoint®, Exchange®, and/or desktop solutions. These protected PDF documents can be read by the AD RMS-aware Foxit Reader.

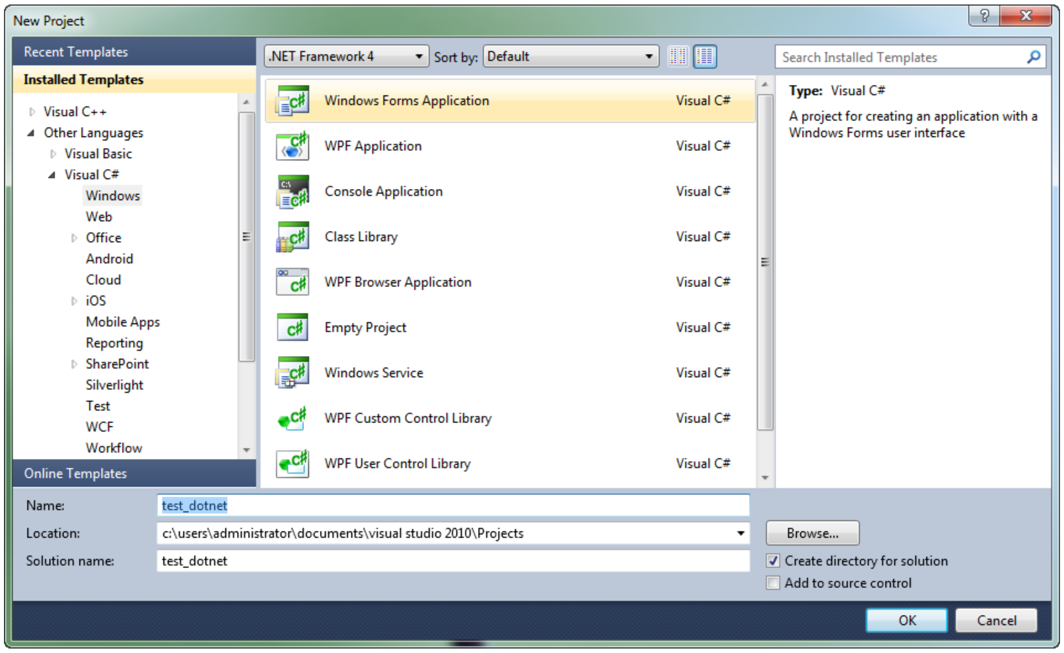
Software Development Kits (SDKs)

Foxit PDF SDKs enable developers and device manufacturers to leverage Foxit's award winning PDF technology in their applications and devices. The SDKs allow developers to incorporate powerful PDF technology to view, search, and annotate PDF. The SDKs provide these capabilities in an easy to integrate, platform independent solution that allows developers to reduce time to market by developing features once and then easily port these capabilities to other platforms.

Figure 3-5

3.4 How to create your own project

In this section, we will show you how to create your own project based on .NET Framework and how to render a PDF document using Foxit PDF SDK. Create a new Windows Forms Application called "test_dotnet" in Visual Studio 2010, and we will be using C# and .NET Framework 4.0, which is shown in Figure 3-6.



.NET Framework 4

Sort by: Default

Type: Visual C#

A project for creating an application with a Windows Forms user interface

Name: test_dotnet

Location: c:\users\administrator\documents\visual studio 2010\Projects

Solution name: test_dotnet

[✓] Create directory for solution

[ ] Add to source control

Figure 3-6

Copy the "lib" folder from the package to the project. To run this project in Visual Studio 2010, please follow the steps below:

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

   i. In Solution Explorer, right-click the **References** and click **Add Reference...** as shown in Figure 3-7.

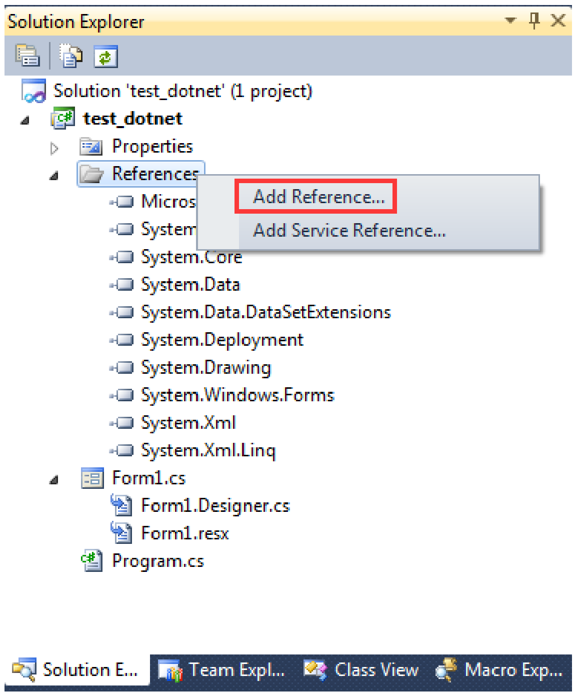


Figure 3-7

ii. In the **Add Reference** dialog, click **Browse**, navigate to the "test_dotnet\lib\framework_40\x64" folder, select **fsdk_dotnet.dll** dynamic library, and then click **OK**. It is shown in Figure 3-8.

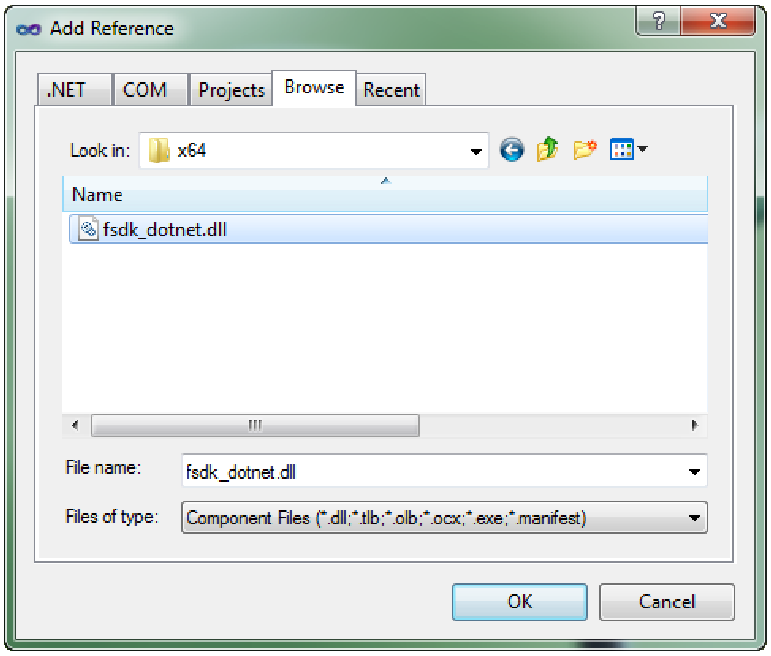
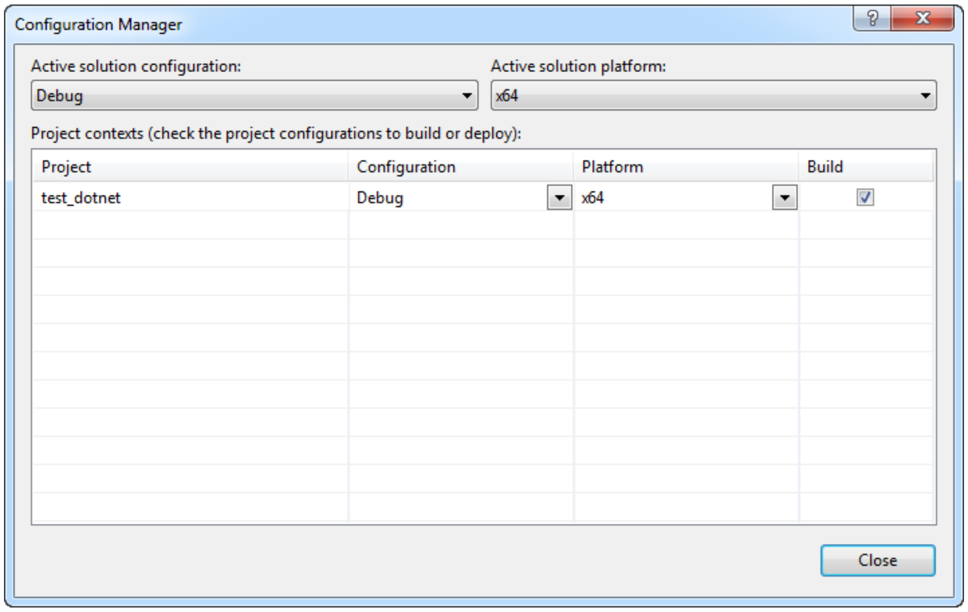

Add Reference
.NET
COM
Projects
Browse
Recent
Look in: x64
Name
fsdk_dotnet.dll
File name: fsdk_dotnet.dll
Files of type: Component Files (*.dll;*.tlb;*.olb;*.ocx;*.exe;*.manifest)
OK
Cancel

Figure 3-8

Note: *Please make sure the platform target (x86 or x64) of the application matches the fsdk_dotnet.dll architecture. Here, we choose the x64 library based on .NET Framework 4.0, so we need to change the build architecture of the project.*

b) Change the build architecture of the project. Click on **Build** -> **Configuration Manager** and select **x64** for the "Active solution platform" as shown in Figure 3-9.



Active solution configuration: Debug
Active solution platform: x64
Project contexts (check the project configurations to build or deploy):

Project	Configuration	Platform	Build
test_dotnet	Debug	x64	✓

Figure 3-9

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

1) Add a button in the Form which is used for choosing a PDF file.

2) Load Foxit PDF SDK library. We do this in the "Program.cs". Open up "**Program.cs**", at first, add using statement to the beginning of the CS file.

```
using Foxit;
```

And then, add the codes as follows:

```
static void Main()
{
        Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault(false);

        // Load the Foxit PDF SDK library.
        string license_id;
        string unlockCode;

        bool bSuccess = Library.Load(license_id, unlockCode);
        if (!bSuccess) return;
        Library.LoadSystemFonts();

        Application.Run(new Form1());
}
```

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 in the "**Form1.cs**".

```
using System.Windows;
using System.IO;
using System.Drawing.Imaging;
using System.Runtime.InteropServices;
using Foxit;
using Foxit.PDF;
```

4) Declare a variable representing a Windows picture box control for displaying an image.

```
private PictureBox pb = null;
```

5) Initialize the variable in the constructor.

```
public Form1()
{
        InitializeComponent();

        pb = new PictureBox();
        this.Controls.Add(pb);
}
```

6) Realize the **button1_Click** event which is used for choosing a PDF file and rendering it.

Choose a PDF file.

```
OpenFileDialog openPDFFileDialog = new OpenFileDialog();
openPDFFileDialog.FileName = "";
openPDFFileDialog.Filter = "*.pdf|*.pdf";
openPDFFileDialog.ShowDialog(this);
```

Open a PDF file.

```
FileStream m_fileStream = new FileStream(openPDFFileDialog.FileName,
FileMode.Open);
```

Load the selected PDF document.

```
Document m_document = new Document();
bool bLoadSuccess = m_document.Load(m_fileStream, "", 0);
```

Load the first page of the PDF document.

```
Page page = m_document.LoadPage(0, 0, null);
```

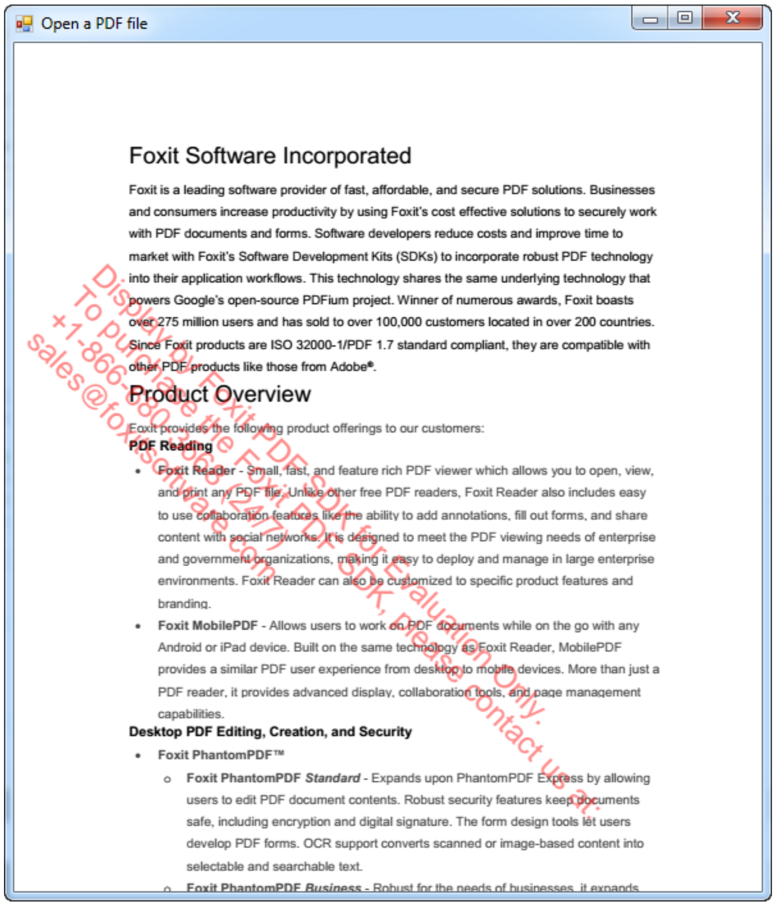
Render the loaded page.

Note: *You should add a reference to the "WindowsBase.dll".*

Add the following codes to render the loaded page.

```
System.Windows.Size size = page.GetSize();
int nWidth = (int)size.Width;
int nHeight = (int)size.Height;

Matrix matrix = page.GetDisplayMatrix(0, 0, nWidth, nHeight, 0);

PixelSource pixSrc = new PixelSource();
pixSrc.Width = nWidth;
pixSrc.Height = nHeight;
pixSrc.Format = PixelFormat.Format32bppRgb;

page.RenderPage(pixSrc, matrix, (uint)RenderFlags.Annot, null);

GCHandle hObject = GCHandle.Alloc(pixSrc.PixelBuffer, GCHandleType.Pinned);
IntPtr pObject = hObject.AddrOfPinnedObject();
int stride = (pixSrc.Width * 32 + 31) / 32 * 4;
Bitmap bitmap = new Bitmap(pixSrc.Width, pixSrc.Height, stride, pixSrc.Format,
pObject);

pb.Image = bitmap;
pb.Size = new System.Drawing.Size(nWidth, nHeight);

bitmap = null;
GC.Collect();
```

d) Build the project by right clicking the project and selecting **Build**. Then, Press **F5** to run the project, and click the button to choose and open a PDF document. The selected PDF document will be displayed as shown in Figure 3-10.



Foxit Software Incorporated

Foxit is a leading software provider of fast, affordable, and secure PDF solutions. Businesses and consumers increase productivity by using Foxit's cost effective solutions to securely work with PDF documents and forms. Software developers reduce costs and improve time to market with Foxit's Software Development Kits (SDKs) to incorporate robust PDF technology into their application workflows. This technology shares the same underlying technology that powers Google's open-source PDFium project. Winner of numerous awards, Foxit boasts over 275 million users and has sold to over 100,000 customers located in over 200 countries. Since Foxit products are ISO 32000-1/PDF 1.7 standard compliant, they are compatible with other PDF products like those from Adobe®.

Product Overview

Foxit provides the following product offerings to our customers:

PDF Reading

- **Foxit Reader** - Small, fast, and feature rich PDF viewer which allows you to open, view, and print any PDF file. Unlike other free PDF readers, Foxit Reader also includes easy to use collaboration features like the ability to add annotations, fill out forms, and share content with social networks. It is designed to meet the PDF viewing needs of enterprise and government organizations, making it easy to deploy and manage in large enterprise environments. Foxit Reader can also be customized to specific product features and branding.
- **Foxit MobilePDF** - Allows users to work on PDF documents while on the go with any Android or iPad device. Built on the same technology as Foxit Reader, MobilePDF provides a similar PDF user experience from desktop to mobile devices. More than just a PDF reader, it provides advanced display, collaboration tools, and page management capabilities.

Desktop PDF Editing, Creation, and Security

- **Foxit PhantomPDF™**
  - **Foxit PhantomPDF *Standard*** - Expands upon PhantomPDF Express by allowing users to edit PDF document contents. Robust security features keep documents safe, including encryption and digital signature. The form design tools let users develop PDF forms. OCR support converts scanned or image-based content into selectable and searchable text.
  - **Foxit PhantomPDF *Business*** - Robust for the needs of businesses, it expands

Figure 3-10

Complete program:

The following codes show **Program.cs** and **Form1.cs** in their entirety.

Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;
using Foxit;

namespace test_dotnet
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);

            // Load the Foxit PDF SDK library.
            // 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 = " ";
            bool bSuccess = Library.Load(license_id, unlockCode);
            if (!bSuccess) return;
            Library.LoadSystemFonts();

            Application.Run(new Form1());
        }
    }
}
```

Form1.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Windows;
using System.IO;
using System.Drawing.Imaging;
using System.Runtime.InteropServices;
using Foxit;
using Foxit.PDF;

namespace test_dotnet
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();

            // Initialize a Windows picture box control for displaying an image.
            pb = new PictureBox();
            this.Controls.Add(pb);
        }

        private PictureBox pb = null;

        private void button1_Click(object sender, EventArgs e)
        {
            // Hide the button.
            this.button1.Visible = false;

            // Choose a PDF file.
            OpenFileDialog openPDFFileDialog = new OpenFileDialog();
            openPDFFileDialog.FileName = "";
            openPDFFileDialog.Filter = "*.pdf|*.pdf";
            openPDFFileDialog.ShowDialog(this);

            // Open a PDF File
            FileStream m_fileStream = new FileStream(openPDFFileDialog.FileName,
FileMode.Open);

            // Load the selected PDF document.
            Document m_document = new Document();
            bool bLoadSuccess = m_document.Load(m_fileStream, "", 0);

            // Load the first page of the PDF document.
            Page page = m_document.LoadPage(0, 0, null);

            // Render the loaded page.
            System.Windows.Size size = page.GetSize();
            int nWidth = (int)size.Width;
            int nHeight = (int)size.Height;
            Matrix matrix = page.GetDisplayMatrix(0, 0, nWidth, nHeight, 0);
            PixelSource pixSrc = new PixelSource();
            pixSrc.Width = nWidth;
            pixSrc.Height = nHeight;
            pixSrc.Format = PixelFormat.Format32bppRgb;

            page.RenderPage(pixSrc, matrix, (uint)RenderFlags.Annot, null);

            GCHandle hObject = GCHandle.Alloc(pixSrc.PixelBuffer,
GCHandleType.Pinned);
            IntPtr pObject = hObject.AddrOfPinnedObject();
            int stride = (pixSrc.Width * 32 + 31) / 32 * 4;
            Bitmap bitmap = new Bitmap(pixSrc.Width, pixSrc.Height, stride,
pixSrc.Format, pObject);

            pb.Image = bitmap;
            pb.Size = new System.Drawing.Size(nWidth, nHeight);

            bitmap = null;
            GC.Collect();

        }
    }
}
```

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 based on .NET Framework 4.0. 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

```
string license_id = " ";
string unlockCode = " ";
bool bSuccess = Foxit.Library.Load(license_id, unlockCode);
if (!bSuccess) return;
Foxit.Library.LoadSystemFonts();
```

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::Load or Foxit::PDF::Document::LoadFromFilePath. 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 Load()
FileStream fileStream = new
FileStream("C:\\Users\\Administrator\\Desktop\\1.pdf", FileMode.Open);
Document pdfDoc = new Document();
Boolean bRet = pdfDoc.Load(fileStream, "", 0);
...
pdfDoc.Close();
fileStream.Close();
```

```
//load a PDF document with LoadFromFilePath()
Document pdfDoc = new Document();
Boolean bRet =
pdfDoc.LoadFromFilePath("C:\\Users\\Administrator\\Desktop\\1.pdf", "", 0);
...
pdfDoc.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.LoadPage(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 PDF file by SaveAs()
//Assuming Document document has been loaded.

FileStream saveFileStream = new
FileStream("C:\\Users\\Administrator\\Desktop\\SavedResult.pdf",
FileMode.Create);
Boolean bRet = pdfDoc.SaveAs(saveFileStream, (uint)SaveFlags.Incremental,
null);
saveFileStream.Close();
```

```
//Save PDF file by SaveToFile()
//Assuming Document document has been loaded.

Boolean bRet =
pdfDoc.SaveToFile("C:\\Users\\Administrator\\Desktop\\SavedResult.pdf",
(uint)SaveFlags.Incremental, null);
```

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, inserting 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 = document.CreateAttachment(attachFileInfo, insertIndex,
true);

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::LoadPage, 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

```
//Retrieve 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 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 URL action
//Assuming Document document has been loaded.
...
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 strem.
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);

FileStream streamSave = new
FileStream("C:\\Users\\Administrator\\Desktop\\SignedResult.pdf", ,
FileMode.Create);
bool bRet = sig.Sign(streamSave, null);
streamSave.Close();
```

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 dotNet?**
   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 dotNet?**
   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 dotNet?**
   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 (Dotnet) 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