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.

Foxit PDF SDK provides high-performance libraries to help any software developer add robust PDF functionality to their enterprise, mobile and cloud applications across all platforms (includes Windows, Mac, Linux, Web, Android, iOS, and UWP), using the most popular development languages and environments.

1.1 Why Foxit PDF SDK is your choice

Foxit is 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. Customers choose Foxit PDF SDK product for the following reasons:

- **Easy to integrate**

Developers can seamlessly integrate Foxit PDF SDK into their own applications.

- **Lightweight footprint**

Do not exhaust system resource and deploys quickly.

- **Cross-platform support**

Support current mainstream platforms, such as Windows, Mac, Linux, Web, Android, iOS, and UWP.

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

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

- **Premium World-side Support**

Foxit offers premium support for its developer products because when you are developing mission critical products you need the best support. Foxit has one of the PDF industry's largest team of support engineers. Updates are released on a regular basis to improve user experience by adding new features and enhancements.

1.2 Foxit PDF SDK for .NET

Application developers who use Foxit PDF SDK can leverage Foxit's powerful, standard-compliant PDF technology to securely display, create, edit, annotate, format, organize, print, share, secure, search documents as well as to fill PDF forms. Additionally, Foxit PDF SDK includes a built-in, embeddable PDF Viewer, making the development process easier and faster. For more detailed information, please visit the website https://developers.foxitsoftware.com/pdf-sdk/.

In this guide, we focus on the introduction of Foxit PDF SDK for Windows platform with .NET framework.

Foxit PDF SDK for .NET is a Component for Windows which ships with simple-to-use APIs that can help .NET developers seamlessly integrate powerful PDF technology into their own projects based on .NET Framework 4.0 or higher. It provides rich features on PDF documents, such as PDF viewing, bookmark navigating, text selecting/copying/searching, PDF signatures, PDF forms, rights management, PDF annotations, and full text search.

1.3 Evaluation

Foxit PDF SDK allows users to download trial version to evaluate SDK. The trial version has no difference from a standard version except for the 10-day limitation trial period and the trail watermarks that will be generated on the PDF pages. After the evaluation period expires, customers should contact Foxit sales team and purchase licenses to continue using Foxit PDF SDK.

1.4 License

Developers should purchase licenses to use Foxit PDF SDK in their solutions. Licenses grant users permissions to release their applications based on PDF SDK libraries. However, users are prohibited to distribute any documents, sample codes, or source codes in the SDK released package to any third party without the permission from Foxit Software Incorporated.

1.5 About this guide

This guide is intended for the developers who need to integrate Foxit PDF .NET SDK into their own applications. It aims at introducing the installation package, and the usage of SDK.

2 GETTING STARTED

It’s very easy to setup Foxit PDF SDK and see it in action! This guide will provide you with a brief introduction on how to integrate Foxit PDF SDK into the projects based on .NET Framework 4.0 or higher. The following sections introduce the contents of system requirements, the installation package as well as how to run a demo, and create your own project.

2.1 System Requirements

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

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

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

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

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

2.2 What is in the package

Download Foxit PDF SDK zip for .NET package and extract it to a new directory "foxitpdfsdk_7_1_win_dotnet", which is shown in Figure 2-1.

NOTE*: the highlighted rectangle in the figure is just the version of Foxit PDF SDK. Here the SDK version is 7.1, so it shows 7_1 in the package name.*

The release package contains the following folders:

doc:	API references, developer guide
examples:	sample projects and demos
lib:	libraries and license files

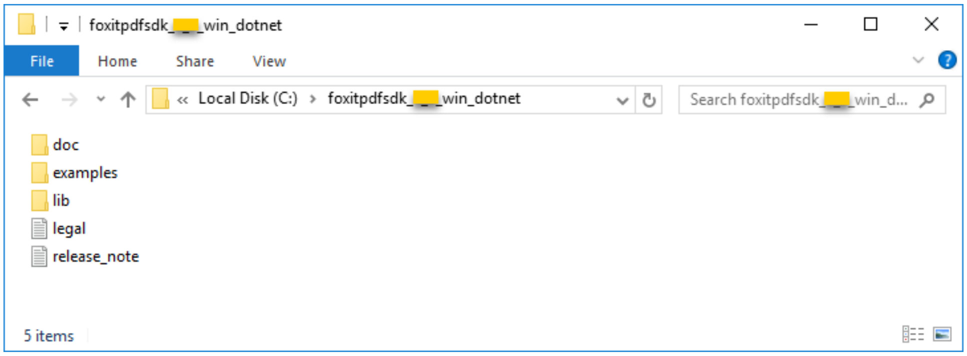


Figure 2-1

In the "examples" folder, there are two types of demos. "\examples\simple_demo" contains more than 30 demos that cover a wide range of PDF applications. "\examples\view_demo" contains a UI demo that realizes a lite PDF viewer.

2.3 How to run a demo

Simple Demo

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

To run a demo in Visual Studio 2010/2015/2017 (except ocr, compliance and html2pdf demos which will be introduced in section 3), you can follow the steps below:

1) Load the visual studio solution file "simple_demo_vs2010.sln" or "simple_demo_vs2015.sln" or "simple_demo_vs2017.sln" (depending on your Visual Studio version) in the "\examples\simple_demo" folder.

2) Build all the demos by clicking "Build > Build Solution". Alternatively, if you merely want to build a specific demo, you can right-click it and then choose "Build" or load the "*.csproj" file in the folder of a specific demo project and then build it.

   After building, the executable file ".exe" will be generated in the "\examples\simple_demo\bin" folder. And the names of the executable files depend on the build configurations.

3) Run a specific executable file, just double-click it. If you want to see the detailed executing processes, you can run it in command line. Start "cmd.exe", navigate to "\examples\simple_demo\bin", and run a specific executable file.

   "\examples\simple_demo\input_files" contains all the input files used among these demos. Some demos will generate output files (pdf, text or image files) to a folder named by the project name under "example\simple_demo\output_files\" folder.

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

   a) To install "foxit.cer", double-click it to start the certificate import wizard. Then select "Install certificate… > Next > Next > Finish".

b) To install "foxit_all.pfx", double-click it to start the certificate import wizard. Then select "Next > Next > (Type the password for the private key in the textbox) and click Next > Next > Finish".

OCR and Compliance demos

For **ocr** and **compliance** demos, you should build a resource directory at first, please contact Foxit support team or sales team to get the resource files packages. For more details about how to run the demos, please refer to section 3.34 "OCR" and section 3.35 "Compliance".

HTML to PDF demo

For html2pdf demo, you should contact Foxit support team or sales team to get the engine files package for converting from HTML to PDF at first. For more details about how to run the demo, please refer to section 3.37 "HTML to PDF Conversion".

View Demo

This view demo provides an example for .NET developers to realize a PDF reader using PDF SDK APIs.

To run the demo in Visual Studio, load "PDFReader_VS2010.sln" or "PDFReader_VS2015.sln" or "PDFReader_VS2017.sln" (depending on your Visual Studio version) in the "\examples\view_demo\PDFReader" folder, and then click "Debug > Start Without Debugging" to run it. After the demo starts, you will see the following window as shown in Figure 2-2.

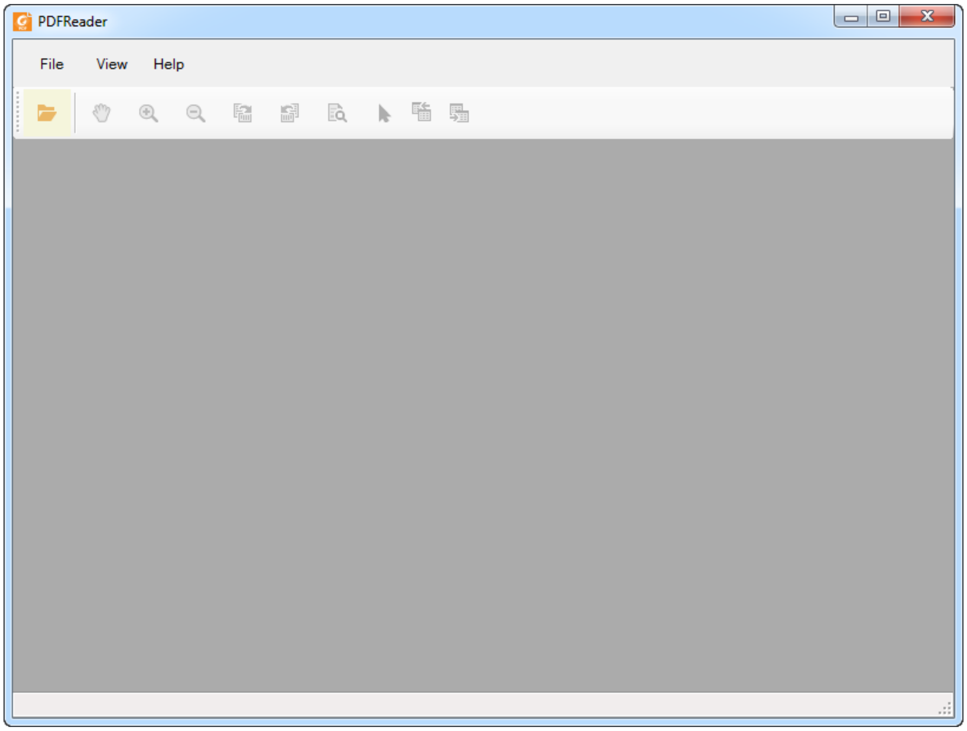
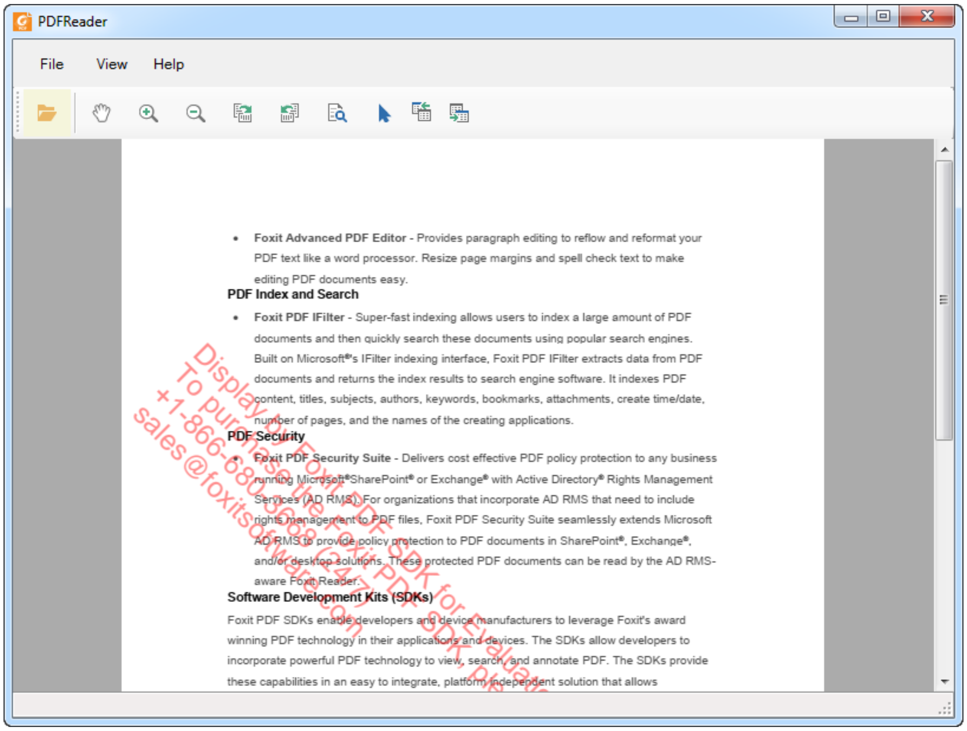


Figure 2-2

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

This demo provides the features like rendering a PDF document, zooming, page rotation, text selection and search, and page turning. For example, click the Next Page button to view the next page, which is shown in Figure 2-3.



- **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

Figure 2-3

2.4 How to create a simple project

In this section, we will show you how to use Foxit PDF .NET SDK to create a simple project that renders the first page of a PDF to a bitmap and saves it as a JPG image. Please follow the steps below:

1) Open Visual Studio and create a new C# Console Application named "test_dotnet".

2) Copy the "**lib**" folder from the "foxitpdfsdk_7_1_win_dotnet" folder to the project "test_dotnet" folder.

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

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

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

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

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

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

5) Add "fsdk.dll" to the project.

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

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

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

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

7) Initialize Foxit PDF SDK library. It is necessary for apps to initialize Foxit PDF SDK using a license before calling any APIs. The trial license files can be found in the "lib" folder.

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

   ***Note** The value of "sn" can be got from "**gsdk_sn.txt**" (the string after "SN=") and the value of "key" can be got from "**gsdk_key.txt**" (the string after "Sign=").*

8) Load a PDF document, and parse the first page of the document. Let us assume that you have already put a "Sample.pdf" to the "test_dotnet\test_dotnet" folder.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

            // The value of "sn" can be got from "gsdk_sn.txt" (the string after "SN=").
            // The value of "key" can be got from "gsdk_key.txt" (the string after "Sign=").
            string sn = " ";
            string key = " ";
            ErrorCode error_code = Library.Initialize(sn, key);
            if (error_code != ErrorCode.e_ErrSuccess)
            {
                return;
            }

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

                using (PDFPage page = doc.GetPage(0))
                {
                    // Parse page.
                    page.StartParse((int)foxit.pdf.PDFPage.ParseFlags.e_ParsePageNormal, null,
false);

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

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

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

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

3 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 or higher. You can refer to the API reference [2] to get more details about the APIs used in all of the examples.

3.1 Initialize Library

It is necessary for applications to initialize Foxit PDF SDK before calling any APIs. The function foxit.common.Library.Initialize is provided to initialize 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.common.Library.Release to release it.

Note: *The parameter "sn" can be found in the "**gsdk_sn.txt**" (the string after "SN=") and the "key" can be found in the "**gsdk_key.txt**" (the string after "Sign=").*

Example:

3.1.1 How to initialize Foxit PDF SDK

```
using foxit.common;

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

3.2 Document

A PDF document object can be constructed with an existing PDF file from file path, memory buffer, a custom implemented ReaderCallback object and an input file stream. Then call function PDFDoc.Load or PDFDoc.StartLoad to load document content. A PDF document object is used for document level operation, such as opening and closing files, getting page, annotation, metadata and etc.

Example:

3.2.1 How to create a PDF document from scratch

```
using foxit.pdf;
...
PDFDoc doc = new PDFDoc();
```

Note*: It creates a new PDF document without any pages.*

3.2.2 How to load an existing PDF document from file path

```
using foxit.pdf;
using foxit.common;
...

PDFDoc doc = new PDFDoc("Sample.pdf");
error_code = doc.Load("");
if (error_code != ErrorCode.e_ErrSuccess) return;
...
```

3.2.3 How to load an existing PDF document from a memory buffer

```
using foxit.pdf;
using foxit.common;
...

byte[] byte_buffer = File.ReadAllBytes(input_file);
IntPtr buffer = System.Runtime.InteropServices.Marshal.AllocHGlobal(byte_buffer.Length);
try {
       System.Runtime.InteropServices.Marshal.Copy(byte_buffer, 0, buffer,
byte_buffer.Length);
}
Finally {
       System.Runtime.InteropServices.Marshal.FreeHGlobal(buffer);
}
PDFDoc doc = new PDFDoc(buffer, (uint)byte_buffer.Length);
error_code = doc.Load("");
if (error_code != ErrorCode.e_ErrSuccess) return;
...
```

3.2.4 How to load an existing PDF document from a file read callback object

```
using foxit.pdf;
using foxit.common;
...

class FileReader : FileReaderCallback
    {
        private FileStream file_ = null;
        private long offset_ = 0;

        public FileReader(long offset)
        {
            this.offset_ = offset;
        }

        public Boolean LoadFile(String file_path)
        {
            file_ = new FileStream(file_path, FileMode.OpenOrCreate);
            return true;
        }

        public override long GetSize()
        {
            return this.offset_;
        }

        public override bool ReadBlock(IntPtr buffer, long offset, uint size)
        {
            int read_size = 0;
            file_.Seek(offset, SeekOrigin.Begin);
            byte[] array = new byte[size + 1];
            read_size = file_.Read(array, 0, (int)size);
            Marshal.Copy(array, 0, buffer, (int)size);
            return read_size == size ? true : false;
        }

        public override void Release()
        {
            this.file_.Close();
        }
    }
...

FileReader file_reader = new FileReader(offset);
file_reader.LoadFile(file_name);

PDFDoc doc_real = new PDFDoc(file_reader, false)

error_code = doc.Load("");
if (error_code != ErrorCode.e_ErrSuccess) return;
...
```

3.2.5 How to load PDF document and get the first page of the PDF document

```
using foxit.pdf;
using foxit.common;
...

PDFDoc doc = new PDFDoc("Sample.pdf");
error_code = doc.Load("");
if (error_code != ErrorCode.e_ErrSuccess) return;
...
// Get the first page of the document.
PDFPage page = doc.GetPage(0);
// Parse page.
page.StartParse((int)foxit.pdf.PDFPage.ParseFlags.e_ParsePageNormal, null, false);
...
```

3.2.6 How to save a PDF to a file

```
using foxit.pdf;
using foxit.common;
...

PDFDoc doc = new PDFDoc("Sample.pdf");
error_code = doc.Load("");
if (error_code != ErrorCode.e_ErrSuccess) return;
...
// Do operations for the PDF document.
...
// Save the changes for the PDF document.
string newPdf = "the output path for the saved PDF";
doc.SaveAs(newPdf, (int)PDFDoc.SaveFlags.e_SaveFlagNoOriginal);
```

3.2.7 How to save a document into memory buffer by FileWriterCallback

```
using foxit.pdf;
using foxit.common;
using foxit.common.fxcrt;
using System.IO;
using System.Runtime.InteropServices;
...

class FileWriter : FileWriterCallback
{
     private MemoryStream memoryfilestream = new MemoryStream();
     public FileWriter()
     {
     }

     public override long GetSize()
     {
         return this.memoryfilestream.Length;
     }

     public override bool WriteBlock(IntPtr pData, long offset, uint size)
     {
         byte[] ys = new byte[size];
         Marshal.Copy(pData, ys, 0, (int)size);
         memoryfilestream.Write(ys, 0, (int)size);
         return true;
     }

     public override bool Flush()
     {
         return true;
     }

     public override void Release()
     {
     }
}
...

FileWriter fileWriter = new FileWriter();

// Assuming PDFDoc doc has been loaded.
...

doc.StartSaveAs(fileWriter, (int) PDFDoc.SaveFlags.e_SaveFlagNoOriginal,null);
...
```

3.3 Page

PDF Page is the basic and important component of PDF Document. A PDFPage object is retrieved from a PDF document by function PDFDoc.GetPage. Page level APIs provide functions to parse, render, edit (includes creating, deleting and flattening) a page, retrieve PDF annotations, read and set the properties of a page, and etc. For most cases, A PDF page needs to be parsed before it is rendered or processed.

Example:

3.3.1 How to get page size

```
using foxit.pdf;
using foxit.common;
...

// Assuming PDFPage page has been loaded and parsed.
...
int width = (int)(page.GetWidth());
int height = (int)(page.GetHeight());
...
```

3.3.2 How to calculate bounding box of page contents

```
using foxit.pdf;
...

// Assuming PDFPage page has been loaded and parsed.
...

RectF ret = page.CalcContentBBox(PDFPage.CalcMarginMode.e_CalcContentsBox);
...
```

3.3.3 How to create a PDF page and set the size

```
using foxit.pdf;
...

// Assuming PDFDoc doc has been loaded.

PDFPage page = doc.InsertPage(index, PageWidth, PageHeight);
```

3.3.4 How to delete a PDF page

```
using foxit.pdf;
...

// Assuming PDFDoc doc has been loaded.

// Remove a PDF page by page index.
doc.RemovePage(index);

// Remove a specified PDF page.
doc.RemovePage(page);
...
```

3.3.5 How to flatten a PDF page

```
using foxit.pdf;
...

// Assuming PDFPage page has been loaded and parsed.
// Flatten all contents of a PDF page.
page.Flatten(true, (int)PDFPage.FlattenOptions.e_FlattenAll);

// Flatten a PDF page without annotations.
page.Flatten(true, (int)PDFPage.FlattenOptions.e_FlattenNoAnnot);

// Flatten a PDF page without form controls.
page.Flatten(true, (int)PDFPage.FlattenOptions.e_FlattenNoFormControl);

// Flatten a PDF page without annotations and form controls (Equals to nothing to be 
flattened).
page.Flatten(true, (int)(PDFPage.FlattenOptions.e_FlattenNoAnnot | 
PDFPage.FlattenOptions.e_FlattenNoFormControl));
...
```

3.3.6 How to get and set page thumbnails in a PDF document

```
using foxit.pdf;
...

// Assuming PDFPage page has been loaded and parsed.

// Get page thumbnails.
page.LoadThumbnail();

// Set thumbnails to the page.
// Assuming Bitmap bitmap has been created.
page.SetThumbnail(bitmap);
...
```

3.4 Render

PDF rendering is realized through the Foxit renderer, a graphic engine that is used to render page to a bitmap or platform graphics device. Foxit PDF SDK provides APIs to set rendering options/flags, for example set flag to decide whether to render form fields and signature, whether to draw image anti-aliasing and path anti-aliasing. To do rendering, you can use the following APIs:

- To render page and annotations, first use function Renderer.SetRenderContentFlags to decide whether to render page and annotation both or not, and then use function Renderer.StartRender to do the rendering. Function Renderer.StartQuickRender can also be used to render page but only for thumbnail purpose.
- To render a single annotation, use function Renderer.RenderAnnot.
- To render on a bitmap, use function Renderer.StartRenderBitmap.
- To render a reflowed page, use function Renderer.StartRenderReflowPage.

Widget annotation is always associated with form field and form control in Foxit PDF SDK. For how to render widget annotations, here is a recommended flow:

- After loading a PDF page, first render the page and all annotations in this page (including widget annotations).
- Then, if use pdf.interform.Filler object to fill the form, the function pdf.interform.Filler.Render should be used to render the focused form control instead of the function Renderer.RenderAnnot.

Example:

3.4.1 How to render a page to a bitmap

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

// Assuming PDFPage page has been loaded and parsed.

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

// Prepare a bitmap for rendering.
System.Drawing.Bitmap bitmap = new System.Drawing.Bitmap(width, height,
System.Drawing.Imaging.PixelFormat.Format32bppArgb);
Graphics draw = Graphics.FromImage(bitmap);
draw.Clear(Color.White);
// Render page.
Renderer render = new Renderer(bitmap, false);
render.StartRender(page, matrix, null);
...
```

3.4.2 How to render page and annotation

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

// Assuming PDFPage page has been loaded and parsed.
...

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

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

```
// Render page
Renderer render = new Renderer(bitmap, false);
render.SetRenderContentFlags((int)Renderer.ContentFlag.e_RenderAnnot |
(int)Renderer.ContentFlag.e_RenderPage);
render.StartRender(page, matrix, null);
...
```

3.5 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 applications APIs to access attachments such as loading attachments, getting attachments, inserting/removing attachments, and accessing properties of attachments.

Example:

3.5.1 How to insert an attachment file into a pdf

```
using foxit;
using foxit.common;
using foxit.common.fxcrt;
using foxit.pdf;
using foxit.pdf.objects;

// Assuming PDFDoc doc has been loaded.


string text_path = "The input path of the attached file you need to insert";
Attachments attachments = new Attachments(doc, new PDFNameTree());
attachment.AddFromFilePath("OriginalAttachmentsInfo", text_path);
...
```

3.5.2 How to remove a specific attachment of a PDF

```
using foxit;
using foxit.common;
using foxit.common.fxcrt;
using foxit.pdf;
using foxit.pdf.objects;

// Assuming PDFDoc doc has been loaded.

...
Attachments attachment = new Attachments(doc, new PDFNameTree());
string strKey = attachment.GetKey(index);
attachment.RemoveEmbeddedFile(strKey);
...
```

3.6 Text Page

Foxit PDF SDK provides APIs to extract, select, search and retrieve text in PDF documents. PDF text contents are stored in TextPage objects which are related to a specific page. TextPage class can be used to retrieve information about text in a PDF page, such as single character, single word, text content within specified character range or rectangle and so on. It also can be used to construct objects of other text related classes to do more operations for text contents or access specified information from text contents:

- To search text in text contents of a PDF page, construct a TextSearch object with TextPage object.
- To access text such like hypertext link, construct a PageTextLinks object with TextPage object.

Example:

3.6.1 How to extract text from a PDF page

```
using foxit.common;
using foxit.pdf;
...

// Assuming PDFPage page has been loaded and parsed.

using (var text_page = new TextPage(page, (int)TextPage.TextParseFlags.e_ParseTextNormal))
{
       int count = text_page.GetCharCount();
       if (count > 0)
          {
              String chars = text_page.GetChars(0, count);
              writer.Write(chars);
           }
}
...
```

3.6.2 How to get the text within a rectangle area in a PDF

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

RectF rect = new RectF(100, 50, 220, 100);
TextPage text_page = new TextPage(page,
(int)foxit.pdf.TextPage.TextParseFlags.e_ParseTextNormal);
String str_text = text_page.GetTextInRect(rect);
...
```

3.7 Text Search

Foxit PDF SDK provides APIs to search text in a PDF document, a XFA document, a text page or in a PDF annotation's appearance. It offers functions to do a text search and get the searching result:

- To specify the searching pattern and options, use functions TextSearch.SetPattern, TextSearch.SetStartPage (only useful for a text search in PDF document), TextSearch.SetEndPage (only useful for a text search in PDF document) and TextSearch.SetSearchFlags.
- To do the searching, use function TextSearch.FindNext or TextSearch.FindPrev.

- To get the searching result, use function TextSearch.GetMatchXXX().

Example:

3.7.1 How to search a text pattern in a page

```
using foxit.common;
using foxit.pdf;
...

// Assuming PDFDoc doc has been loaded.

using (TextSearch search = new TextSearch(doc, null))
{
     int start_index = 0;
     int end_index = doc.GetPageCount() - 1;
     search.SetStartPage(0);
     search.SetEndPage(doc.GetPageCount() - 1);

     String pattern = "Foxit";
     search.SetPattern(pattern);

     Int32 flags = (int)TextSearch.SearchFlags.e_SearchNormal;
     search.SetSearchFlags(flags);
     int match_count = 0;
     while (search.FindNext())
     {
          RectFArray rect_array = search.GetMatchRects();
          match_count++;
     }
...
```

3.8 Text Link

In a PDF page, some text contents that represent a hypertext link to a website or a resource on the intent, or an email address are the same with common texts. Prior to text link processing, user should first call PageTextLinks.GetTextLink to get a textlink object.

Example:

3.8.1 How to retrieve hyperlinks in a PDF page

```
using foxit.common;
using foxit.pdf;
...

// Assuming PDFPage page has been loaded and parsed.

// Get the text page object.
TextPage text_page = new TextPage(page,
(int)foxit.pdf.TextPage.TextParseFlags.e_ParseTextNormal);
```

```
PageTextLinks page_textlinks = new PageTextLinks(text_page);
TextLink text_link = page_textlinks.GetTextLink(index); // specify an index.
string str_url = text_link.GetURI();
...
```

3.9 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.PDFDoc.GetRootBookmark 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 call function Bookmark.GetFirstChild.

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

- To access the parent bookmark, use function Bookmark.GetParent.
- To access the first child bookmark, use function Bookmark.GetFirstChild.
- To access the next sibling bookmark, use function Bookmark.GetNextSibling.
- To insert a new bookmark, use function Bookmark.Insert.
- To move a bookmark, use function Bookmark.MoveTo.

Example:

3.9.1 How to find and list all bookmarks of a PDF

```
using foxit;
using foxit.common;
using foxit.common.fxcrt;
using foxit.pdf;
using foxit.pdf.actions;
...

//Assuming PDFDoc doc has been loaded.
...
Bookmark root = doc.GetRootBookmark();
Bookmark first_bookmark = root.GetFirstChild();
if (first_bookmark != null)
{
       TraverseBookmark(first_bookmark, 0);
}

Private void TraverseBookmark(Bookmark root, int iLevel)
{
       if (root != null)
       {
              Bookmark child = root.GetFirstChild();
```

```
                while (child != null)
                {
                   TraverseBookmark(child, iLevel + 1);
                   child = child.GetNextSibling();
                }
          }
}
...
```

3.10 Form (AcroForm)

PDF currently supports two different forms for gathering information interactively from the user - AcroForms and XFA forms. Acroforms are the original PDF-based fillable forms, based on the PDF architecture. Foxit PDF SDK provides APIs to view and edit form field programmatically. Form fields are commonly used in PDF documents to gather data. The Form class offers functions to retrieve form fields or form controls, import/export form data and other features, for example:

- To retrieve form fields, please use functions Form.GetFieldCount and Form.GetField.
- To retrieve form controls from a PDF page, please use functions Form.GetControlCount and Form.GetControl.
- To import form data from an XML file, please use function Form.ImportFromXML; to export form data to an XML file, please use function Form.ExportToXML.
- To retrieve form filler object, please use function Form.GetFormFiller.

To import form data from a FDF/XFDF file or export such data to a FDF/XFDF file, please refer to functions pdf.PDFDoc.ImportFromFDF and pdf.PDFDoc.ExportToFDF.

Example:

3.10.1 How to load the forms in a PDF

```
using foxit;
using foxit.common;
using foxit.common.fxcrt;
using foxit.pdf;
using foxit.pdf.interform;
using foxit.pdf.annots;
using foxit.pdf.actions;
...

// Assuming PDFDoc doc has been loaded.

Boolean hasForm = doc.HasForm();
If(hasForm)
       Form form = new Form(doc);
...
```

3.10.2 How to count form fields and get the properties

```
using foxit;
using foxit.common;
using foxit.common.fxcrt;
using foxit.pdf;
using foxit.pdf.interform;
using foxit.pdf.annots;
using foxit.pdf.actions;
...

// Assuming PDFDoc doc has been loaded.

Boolean hasForm = doc.HasForm();
If(hasForm)
       Form form = new Form(doc);
int count = form.GetFieldCount("");
for (int i = 0; i < count; i++)
{
    Field field = form.GetField(i, "");
    ...
}
```

3.10.3 How to export the form data in a PDF to a XML file

```
using foxit;
using foxit.common;
using foxit.common.fxcrt;
using foxit.pdf;
using foxit.pdf.interform;
using foxit.pdf.annots;
using foxit.pdf.actions;
...

// Assuming PDFDoc doc has been loaded.

Boolean hasForm = doc.HasForm();
If(hasForm)
       Form form = new Form(doc);
...
form.ExportToXML(XMLFilePath);
...
```

3.10.4 How to import form data from a XML file

```
using foxit;
using foxit.common;
using foxit.common.fxcrt;
using foxit.pdf;
using foxit.pdf.interform;
using foxit.pdf.annots;
using foxit.pdf.actions;
...

// Assuming PDFDoc doc has been loaded.

Boolean hasForm = doc.HasForm();
```

```
If(hasForm)
       Form form = new Form(doc);
...
form.ImportFromXML(XMLFilePath);
...
```

3.10.5 How to get and set the properties of form fields

```
using foxit;
using foxit.common;
using foxit.common.fxcrt;
using foxit.pdf;
using foxit.pdf.interform;
using foxit.pdf.annots;
using foxit.pdf.actions;
...

// Assuming PDFDoc doc has been loaded.

Boolean hasForm = doc.HasForm();
If(hasForm)
       Form form = new Form(doc);
Field field = form.GetField(0, "Text Field0");
field.GetAlignment();
field.GetAlternateName();
field.SetAlignment(Alignment.e_AlignmentLeft);
field.SetValue("3");
...
```

3.11 XFA Form

XFA (XML Forms Architecture) forms are XML-based forms, wrapped inside a PDF. The XML Forms Architecture provides a template-based grammar and a set of processing rules that allow uses to build interactive forms. At its simplest, a template-based grammar defines fields in which a user provides data.

Foxit PDF SDK provides APIs to render the XFA form, fill the form, export or import form's data.

Note:

- *Foxit PDF SDK provides two callback classes foxit.addon.xfa.AppProviderCallback and foxit.addon.xfa.DocProviderCallback to represent the callback objects as an XFA document provider and an XFA application provider respectively. All the functions in those classes are used as callback functions. Pure virtual functions should be implemented by users.*

- *To use the XFA form feature, please make sure the license key has the permission of the 'XFA' module*

Example:

3.11.1 How to load XFADoc and represent an Interactive XFA form

```
using foxit;
using foxit.common;
using foxit.common.fxcrt;
using foxit.pdf;
using foxit.addon;
using foxit.addon.xfa;
...

// Implement from AppProviderCallback
CFS_XFAAppHandler pXFAAppHandler = new CFS_XFAAppHandler();
Library.RegisterXFAAppProviderCallback(pXFAAppHandler);
string input_file = input_path + "xfa_dynamic.pdf";
using (PDFDoc doc = new PDFDoc(input_file))
{
    error_code = doc.Load(null);
    if (error_code != ErrorCode.e_ErrSuccess)
    {
        Console.WriteLine("The PDFDoc [{0}] Error: {1}\n", input_file, error_code);
        Library.Release();
        return;
    }

    // Implement from DocProviderCallback
    CFS_XFADocHandler pXFADocHandler = new CFS_XFADocHandler();
    using (XFADoc xfa_doc = new XFADoc(doc, pXFADocHandler))
    {
    ...
    }
}
```

3.11.2 How to export and import XFA form data

```
using foxit;
using foxit.common;
using foxit.common.fxcrt;
using foxit.pdf;
using foxit.addon;
using foxit.addon.xfa;
...

// Assuming FSXFADoc xfa_doc has been loaded.

...
xfa_doc.ExportData("xfa_form.xml", XFADoc.ExportDataType.e_ExportDataTypeXML);

xfa_doc.ResetForm();
doc.SaveAs("xfa_dynamic_resetform.pdf", (int)foxit.pdf.PDFDoc.SaveFlags.e_SaveFlagNormal);

xfa_doc.ImportData(output_path + "xfa_form.xml");
doc.SaveAs("xfa_dynamic_importdata.pdf", (int)foxit.pdf.PDFDoc.SaveFlags.e_SaveFlagNormal);
...
```

3.12 Form Filler

Form filler is the most commonly used feature for users. Form filler allows applications to fill forms dynamically. The key point for applications to fill forms is to construct some callback functions for PDF

SDK to call. To fill the form, please construct a Filler object by current Form object or retrieve the Filler object by function Form.GetFormFiller if such object has been constructed. (There should be only one form filler object for an interactive form). For how to fill a form with form filler, you can refer to the view demo "**view_demo**" in the "\examples\view_demo" folder of the download package.

3.13 Form Design

Fillable PDF forms (AcroForm) are especially convenient for preparation of various applications, such as taxes and other government forms. Form design provides APIs to add or remove form fields (Acroform) to or from a PDF file. Designing a form from scratch allows developers to create the exact content and layout of the form they want.

Example:

3.13.1 How to add a text form field to a PDF

```
using foxit;
using foxit.common;
using foxit.common.fxcrt;
using foxit.pdf;
using foxit.pdf.interform;
using foxit.pdf.annots;
using foxit.pdf.actions;
...

// Assuming PDFDoc doc has been loaded.

Control control = form.AddControl(page, "Text Field0", Field.Type.e_TypeTextField, new
RectF(50f, 600f, 90f, 640f))
...
```

3.13.2 How to remove a text form field from a PDF

```
using foxit;
using foxit.common;
using foxit.common.fxcrt;
using foxit.pdf;
using foxit.pdf.interform;
using foxit.pdf.annots;
using foxit.pdf.actions;
...

// Assuming PDFDoc doc has been loaded.

Field field = form.GetField(0, "Text Field0");
form.RemoveField(field);
......
```

3.14 Annotations

3.14.1 General

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 3-1. 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 3-1 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 3-1

Annotation type	Description	Markup	Supported by SDK
Text(Note)	Text annotation	Yes	Yes
Link	Link Annotations	No	Yes
FreeText (TypeWriter/TextBox/Callout)	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
Caret	Caret annotation	Yes	Yes
Ink(pencil)	Ink annotation	Yes	Yes
Popup	Popup annotation	No	Yes
File Attachment	FileAttachment annotation	Yes	Yes
Sound	Sound annotation	Yes	No
Movie	Movie annotation	No	No
Widget*	Widget annotation	No	Yes
Screen	Screen annotation	No	Yes
PrinterMark	PrinterMark annotation	No	No

TrapNet	Trap network annotation	No	No
Watermark*	Watermark annotation	No	Yes
3D	3D annotation	No	No
Redact	Redact annotation	Yes	Yes

Note:

1. The annotation types of widget and watermark are special. They aren't supported in the module of 'Annotation'. The type of widget is only used in the module of 'form filler' and the type of watermark only in the module of 'watermark'.

2. 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:

3.14.1.1 How to add a link annotation to a PDF page

```
using foxit;
using foxit.common;
using foxit.common.fxcrt;
using foxit.pdf;
using foxit.pdf.graphics;
using foxit.pdf.annots;
using foxit.pdf.actions;
...

// Assuming PDFPage page has been loaded and parsed.

Annot annot = page.AddAnnot(Annot.Type.e_Link, new RectF(350, 350, 380, 400))
Link link = new Link(annot);
link.SetHighlightingMode(Annot.HighlightingMode.e_HighlightingToggle);

// Appearance should be reset.
link.ResetAppearanceStream();
...
```

3.14.1.2 How to add a highlight annotation to a page and set the related annotation properties

```
using foxit;
using foxit.common;
using foxit.common.fxcrt;
using foxit.pdf;
using foxit.pdf.graphics;
using foxit.pdf.annots;
using foxit.pdf.actions;
...

// Assuming PDFPage page has been loaded and parsed.

// Add highlight annotation.
Annot annot = page.AddAnnot(Annot.Type.e_Highlight, new RectF(10, 450, 100, 550));
Highlight highlight = new Highlight(annot);
highlight.SetContent("Highlight");
```

```
QuadPoints quad_points = new QuadPoints();
quad_points.first = new foxit.common.fxcrt.PointF(10, 500);
quad_points.second = new foxit.common.fxcrt.PointF(90, 500);
quad_points.third = new foxit.common.fxcrt.PointF(10, 480);
quad_points.fourth = new foxit.common.fxcrt.PointF(90, 480);
QuadPointsArray quad_points_array = new QuadPointsArray();
quad_points_array.Add(quad_points);
highlight.SetQuadPoints(quad_points_array);
highlight.SetSubject("Highlight");
highlight.SetTitle("Foxit SDK");
highlight.SetCreationDateTime(GetLocalDateTime());
highlight.SetModifiedDateTime(GetLocalDateTime());
highlight.SetUniqueID(RandomUID());

// Appearance should be reset.
highlight.ResetAppearanceStream();
...
```

3.14.1.3 How to set the popup information when creating markup annotations

```
using foxit;
using foxit.common;
using foxit.common.fxcrt;
using foxit.pdf;
using foxit.pdf.graphics;
using foxit.pdf.annots;
using foxit.pdf.actions;
...

// Assuming PDFPage page has been loaded and parsed.
// Assuming the annnots in the page have been loaded.

// Create a new note annot and set the properties for it.
Annot annot = page.AddAnnot(Annot.Type.e_Note, new RectF(10, 350, 50, 400));
Note note = new Note(annot);
note.SetIconName("Comment");
note.SetSubject("Note");
note.SetTitle("Foxit SDK");
note.SetContent("Note annotation.");
note.SetCreationDateTime(GetLocalDateTime());
note.SetModifiedDateTime(GetLocalDateTime());
note.SetUniqueID(RandomUID());

// Add popup to note annotation.
Annot annot_popup = page.AddAnnot(Annot.Type.e_Popup, new RectF(300, 450, 500, 550));
Popup popup = new Popup(annot);
popup.SetBorderColor(0x00FF00);
popup.SetOpenStatus(false);
popup.SetModifiedDateTime(GetLocalDateTime());
note.SetPopup(popup);

// Appearance should be reset.
note.ResetAppearanceStream();
...
```

3.14.1.4 How to get a specific annotation in a PDF using device coordinates

```
using foxit;
using foxit.common;
using foxit.pdf;
using foxit.pdf.graphics;
using foxit.pdf.annots;
```

```
...

// Assuming PDFDoc doc has been loaded.
// Assuming PDFPage page has been loaded and parsed.
...

int width = (int)page.GetWidth();
int height = (int)page.GetHeight();

Matrix2D matrix = page.GetDisplayMatrix(0, 0, width, height, page.GetRotation());

// Assuming PointF point has been got.

float tolerance = 3.0f;

Annot annot = page.GetAnnotAtDevicePoint(point, tolerance, matrix);
...
```

3.14.1.5 *How to extract the texts under text markup annotations*

```
using foxit.common;
using foxit.pdf;
using foxit.pdf.annots;
...

// Assuming PDFDoc doc has been loaded.
...

using (var page = doc.GetPage(0))
{
    // Parse the first page.
    page.StartParse((int) PDFPage.ParseFlags.e_ParsePageNormal, null, false);
    // Get a TextPage object.
    using (var text_page = new TextPage(page, (int)
TextPage.TextParseFlags.e_ParseTextNormal))
    {
        int annot_count = page.GetAnnotCount();
        for (int i = 0; i<annot_count; i++)
        {
            Annot annot = page.GetAnnot(i);
TextMarkup text_markup = new TextMarkup(annot);
            if (!text_markup.IsEmpty())
            {
                // Get the texts which intersect with a text markup annotation.
                string text = text_page.GetTextUnderAnnot(text_markup);
            }
        }
    }
}
```

3.14.2 Import annotations from or export annotations to a FDF file

In Foxit PDF SDK, annotations can be created with data not only from applications but also from FDF files. At the same time, PDF SDK supports to export annotations to FDF files.

Example:

3.14.2.1 How to load annotations from a FDF file and add them into the first page of a given PDF

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

// Assuming PDFDoc doc has been loaded.
...

string fdf_file = "The FDF file path";
foxit.fdf.FDFDoc fdf_doc = new foxit.fdf.FDFDoc(fdf_file);
pdf_doc.ImportFromFDF(fdf_doc, (int)foxit.pdf.PDFDoc.DataType.e_Annots, new Range());
...
```

3.15 Image Conversion

Foxit PDF SDK provides APIs for conversion between PDF files and images. Applications could easily fulfill functionalities like image creation and image conversion which supports the following image formats: BMP, TIFF, PNG, JPX, JPEG, and GIF. Foxit PDF SDK can make the conversion between PDF files and the supported image formats except for GIF. It only supports converting GIF images to PDF files.

Example:

3.15.1 How to convert PDF pages to bitmap files

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

// Assuming PDFDoc doc has been loaded.
...

Image image = new Image();

// Get page count
int nPageCount = doc.GetPageCount();
for(int i=0;i<nPageCount;i++)
{
   using (PDFPage page = doc.GetPage(i))
   {
      // Parse page.
      page.StartParse((int)foxit.pdf.PDFPage.ParseFlags.e_ParsePageNormal, null, false);

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

      // Prepare a bitmap for rendering.
      System.Drawing.Bitmap bitmap = new System.Drawing.Bitmap(width, height,
System.Drawing.Imaging.PixelFormat.Format32bppArgb);
      Graphics draw = Graphics.FromImage(bitmap);
      draw.Clear(Color.White);
      // Render page
      Renderer render = new Renderer(bitmap, false);
      render.StartRender(page, matrix, null);
      image.AddFrame(bitmap);
      }
}
...
```

3.15.2 How to convert an image file to PDF file

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

// Assuming PDFDoc doc has been loaded.
...
Image image = new Image(input_file);
int count = image.GetFrameCount();

for (int i = 0; i < count; i++) {
    PDFPage page = doc.InsertPage(i, PDFPage.Size.e_SizeLetter);
    page.StartParse((int)PDFPage.ParseFlags.e_ParsePageNormal, null, false);

    // Add image to page
    page.AddImage(image, i, new PointF(0, 0), page.GetWidth(), page.GetHeight(), true);
}

doc.SaveAs(output_file, (int)PDFDoc.SaveFlags.e_SaveFlagNoOriginal);
...
```

3.16 Watermark

Watermark is a type of PDF annotation and is widely used in PDF document. Watermark is a visible embedded overlay on a document consisting of text, a logo, or a copyright notice. The purpose of a watermark is to identify the work and discourage its unauthorized use. Foxit PDF SDK provides APIs to work with watermark, allowing applications to create, insert, release and remove watermarks.

Example:

3.16.1 How to create a text watermark and insert it into the first page

```
using foxit.common;
using foxit.pdf;
...

// Assuming PDFDoc doc has been loaded.

WatermarkSettings settings = new WatermarkSettings();
settings.offset_x = 0;
settings.offset_y = 0;
settings.opacity = 90;
settings.position = Position.e_PosTopRight;
settings.rotation = -45.0f;
settings.scale_x = 1.0f;
settings.scale_y = 1.0f;

WatermarkTextProperties text_properties = new WatermarkTextProperties();
text_properties.alignment = Alignment.e_AlignmentCenter;
text_properties.color = 0xF68C21;
text_properties.font_style = WatermarkTextProperties.FontStyle.e_FontStyleNormal;
text_properties.line_space = 1;
text_properties.font_size = 12.0f;
text_properties.font = font;

Watermark watermark = new Watermark(doc, "Foxit PDF SDK\nwww.foxitsoftware.com",
text_properties, settings);
watermark.InsertToPage(doc.GetPage(0));

// Save document to file
...
```

3.16.2 How to create an image watermark and insert it into the first page

```
using foxit.common;
using foxit.pdf;
...

// Assuming PDFDoc doc has been loaded.
// Assuming PDFPage page has been loaded and parsed.

WatermarkSettings settings = new WatermarkSettings();
settings.flags = (int)(WatermarkSettings.Flags.e_FlagASPageContents |
WatermarkSettings.Flags.e_FlagOnTop);
settings.offset_x = 0.0f;
settings.offset_y = 0.0f;
settings.opacity = 20;
settings.position = Position.e_PosCenter;
settings.rotation = 0.0f;

Image image = new Image(image_file);
System.Drawing.Bitmap bitmap = image.GetFrameBitmap(0);

settings.scale_x = page.GetWidth() * 0.618f / bitmap.Width;
settings.scale_y = settings.scale_x;

Watermark watermark = new Watermark(doc, image, 0, settings);
watermark.InsertToPage(doc.GetPage(0));

// Save document to file.
...
```

3.16.3 How to remove all watermarks from a page

```
using foxit.common;
using foxit.pdf;
...

// Assuming PDFPage page has been loaded and parsed.
...
```

```
page.RemoveAllWatermarks();
...
// Save document to file
...
```

3.17 Barcode

A barcode is an optical machine-readable representation of data relating to the object to which it is attached. Originally barcodes systematically represented data by varying the widths and spacing of parallel lines, and may be referred to as linear or one-dimensional (1D). Later they evolved into rectangles, dots, hexagons and other geometric patterns in two dimensions (2D). Although 2D systems use a variety of symbols, they are generally referred to as barcodes as well. Barcodes originally were scanned by special optical scanners called barcode readers. Later, scanners and interpretive software became available on devices including desktop printers and smartphones. Foxit PDF SDK provides applications to generate a barcode bitmap from a given string. The barcode types that Foxit PDF SDK supports are listed in Table 3-2.

Table 3-2

Barcode Type	Code39	Code128	EAN8	UPCA	EAN13	ITF	PDF417	QR
Dimension	1D	1D	1D	1D	1D	1D	2D	2D

Example:

3.17.1 How to generate a barcode bitmap from a string

```
using foxit;
using foxit.common;
using foxit.common.fxcrt;
...

// Strings used as barcode content.
String sz_code_str = "TEST-SHEET";

// Barcode format types.
Barcode.Format sz_code_format = Barcode.Format.e_FormatCode39;

//Format error correction level of QR code.
Barcode.QRErrorCorrectionLevel qr_level =
Barcode.QRErrorCorrectionLevel.e_QRCorrectionLevelLow;

//Image names for the saved image files for QR code.
String sz_bmp_qr_name = "/QR_CODE_TestForBarcodeQrCode_L.bmp";

// Unit width for barcode in pixels, preferred value is 1-5 pixels.
int unit_width = 2;

// Unit height for barcode in pixels, preferred value is >= 20 pixels.
int unit_height = 120;

Barcode barcode;
Barcode barcode = new Barcode();
```

```
System.Drawing.Bitmap bitmap = barcode.GenerateBitmap(sz_code_str, sz_code_format, unit_width,
unit_height, qr_level);
...
```

3.18 Security

Foxit PDF SDK provides a range of encryption and decryption functions to meet different level of document security protection. Users can use regular password encryption and certificate-driven encryption, or using their own security handler for custom security implementation. It also provides APIs to integrate with the third-party security mechanism (Microsoft RMS). These APIs allow developers to work with the Microsoft RMS SDK to both encrypt (protect) and decrypt (unprotect) PDF documents.

Note*: For more detailed information about the RMS encryption and decryption, please refer to the simple demo "****security****" in the* "\examples\simple_demo" folder of the download package.

Example:

3.18.1 How to encrypt a PDF file with user password "123" and owner password "456"

```
using foxit.common;
using foxit.pdf;
...
using (var handler = new StdSecurityHandler())
{
     using (var encrypt_data = new StdEncryptData(true, -4,
SecurityHandler.CipherType.e_CipherAES, 16))
     {
          handler.Initialize(encrypt_data, “123”, “456”);
          doc.SetSecurityHandler(handler);

          doc.SaveAs(output_file, (int)PDFDoc.SaveFlags.e_SaveFlagNormal);
     }
}
...
```

3.18.2 How to encrypt a PDF file with Certificate

```
using foxit.common;
using foxit.pdf;
...
PDFDoc doc = new PDFDoc(input_file);
ErrorCode error_code = doc.Load(null);
if (error_code != ErrorCode.e_ErrSuccess){
     return;
}

// Do encryption.
string cert_file_path = input_path + "foxit.cer";

var cms = new EnvelopedCms(new ContentInfo(seed));
cms.ContentEncryptionAlgorithm.Oid.Value = "1.2.840.113549.3.4";
cms.Encrypt(new CmsRecipient(new X509Certificate2(cert_file_path)));
```

```
byte[] bytes = cms.Encode();

byte[] data = new byte[20 + bytes.Length];
Array.Copy(seed, 0, data, 0, 20);
Array.Copy(bytes, 0, data, 20, bytes.Length);
SHA1 sha1 = new SHA1CryptoServiceProvider();
byte[] initial_key = new byte[16];
Array.Copy(sha1.ComputeHash(data), initial_key, initial_key.Length);

byte[][] bytes_array = new byte[1][];
bytes_array[0] = bytes;
var handler = new CertificateSecurityHandler()
var encrypt_data = new CertificateEncryptData()
encrypt_data.Set(true, SecurityHandler.CipherType.e_CipherAES, bytes_array);
handler.Initialize(encrypt_data, initial_key);

doc.SetSecurityHandler(handler);
doc.SaveAs(output_file, (int)PDFDoc.SaveFlags.e_SaveFlagNormal);
...
```

3.18.3 How to encrypt a PDF file with Foxit DRM

```
using foxit.common;
using foxit.pdf;

...
PDFDoc doc = new PDFDoc(input_file);
ErrorCode error_code = doc.Load(null);
if (error_code != ErrorCode.e_ErrSuccess){
     return;
}

// Do encryption.
var handler = new DRMSecurityHandler();
var encrypt_data = new DRMEncryptData(true,"Simple-DRM-filter",
SecurityHandler.CipherType.e_CipherAES, 16, true, -4);
string file_id = "Simple-DRM-file-ID";
string initialize_key = "Simple-DRM-initialize-key";
handler.Initialize(encrypt_data, file_id, initialize_key);
doc.SetSecurityHandler(handler);
doc.SaveAs(output_file, (int)PDFDoc.SaveFlags.e_SaveFlagNormal);
...
```

3.19 Reflow

Reflow is a function that rearranges page content when the page size changes. It is useful for applications that have output devices with difference sizes. Reflow frees the applications from considering layout for different devices. This function provides APIs to create, render, release and access properties of 'reflow' pages.

Example:

3.19.1 How to create a reflow page and render it to a bmp file.

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

// Assuming PDFDoc doc has been loaded.

PDFPage page = doc.GetPage(0);

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

ReflowPage reflow_page = new ReflowPage(page);

// Set some arguments used for parsing the reflow page.
reflow_page.SetLineSpace(0);
reflow_page.SetZoom(100);
reflow_page.SetParseFlags((int)ReflowPage.Flags.e_WithImage);

// Parse reflow page.
reflow_page.StartParse(null);


// Get actual size of content of reflow page. The content size does not contain the margin.
float content_width = reflow_page.GetContentWidth();
float content_height = reflow_page.GetContentHeight();

// Assuming Bitmap bitmap has been created.

// Render reflow page.
Renderer renderer = new Renderer(bitmap, false);
Matrix2D matrix = reflow_page.GetDisplayMatrix(0, 0);
renderer.StartRenderReflowPage(reflow_page, matrix, null);
...
```

3.20 Asynchronous PDF

Asynchronous PDF technique is a way to access PDF pages without loading the whole document when it takes a long time. It's especially designed for accessing PDF files on internet. With asynchronous PDF technique, applications do not have to wait for the whole PDF file to be downloaded before accessing it. Applications can open any page when the data of that page is available. It provides a convenient and efficient way for web reading applications. For how to open and parse pages with asynchronous mode, you can refer to the simple demo "**async_load**" in the "\examples\simple_demo" folder of the download package.

3.21 Pressure Sensitive Ink

Pressure **S**ensitive **I**nk (**PSI**) is a technique to obtain varying electrical outputs in response to varying pressure or force applied across a layer of pressure sensitive devices. In PDF, PSI is usually used for hand writing signatures. PSI data are collected by touching screens or handwriting on boards. PSI data contains coordinates and canvas of the operating area which can be used to generate appearance of PSI. Foxit PDF SDK allows applications to create PSI, access properties, operate on ink and canvas, and release PSI.

Example:

3.21.1 How to create a PSI and set the related properties for it

```
using foxit;
using foxit.common;
using foxit.common.fxcrt;
using foxit.pdf;
using foxit.pdf.annots;
...

PSI psi = new PSI(480, 180, true);

// Set ink diameter.
psi.SetDiameter(9);

// Set ink color.
psi.SetColor(0x434236);

// Set ink opacity.
psi.SetOpacity(0.8f);

// Add points to pressure sensitive ink.
float x = 121.3043f;
float y = 326.6846f;
float pressure = 0.0966f;
Path.PointType type = Path.PointType.e_TypeMoveTo;
foxit.common.fxcrt.PointF pt = new foxit.common.fxcrt.PointF(x, y);
psi.AddPoint(pt, type, pressure);
...
```

3.22 Wrapper

Wrapper provides a way for users to save their own data related to a PDF document. For example, when opening an encrypted unauthorized PDF document, users may get an error message. In this case, users can still access wrapper data even when they do not have permissions to the PDF content. The wrapper data could be used to provide information like where to get decryption method of this document.

Example:

3.22.1 How to open a document including wrapper data.

```
using foxit;
using foxit.common;
using foxit.common.fxcrt;
using foxit.pdf;
using foxit.pdf.graphics;
using foxit.pdf.objects;

// Assuming PDFDoc doc has been loaded.

// file_name is PDF document which includes wrapper data.
PDFDoc doc = new PDFDoc(file_name);
ErrorCode code = doc.Load(null);
if (code != ErrorCode.e_ErrSuccess)
```

```
{
        return;
}
if (!doc.IsWrapper())
{
        return;
}

long offset = doc.GetWrapperOffset();
FileReader file_reader = new FileReader(offset);
file_reader.LoadFile(file_name);
...
```

3.23 PDF Objects

There are eight types of object in PDF: Boolean object, numerical object, string object, name object, array object, dictionary object, stream object and null object. PDF objects are document level objects that are different from page objects (see 3.24) which are associated with a specific page each. Foxit PDF SDK provides APIs to create, modify, retrieve and delete these objects in a document.

Example:

3.23.1 How to remove some properties from catalog dictionary

```
using foxit;
using foxit.common;
using foxit.common.fxcrt;
using foxit.pdf;
using foxit.pdf.graphics;
using foxit.pdf.objects;
...

// Assuming PDFDoc doc has been loaded.

PDFDictionary doc_catalog_dict = doc.GetCatalog();
String[] key_strings = { "Type", "Boolean", "Name", "String", "Array", "Dict" };
int count = key_strings.Length;
for (int i = 0; i < count; i++)
{
   if (catalog.HasKey(key_strings[i]))
   {
       catalog.RemoveAt(key_strings[i]);
   }
}
...
```

3.24 Page Object

Page object is a feature that allows novice users having limited knowledge of PDF objects (see 3.23 for details of PDF objects) to be able to work with text, path, image, and canvas objects. Foxit PDF SDK provides APIs to add and delete PDF objects in a page and set specific attributes. Using page object, users can create PDF page from object contents. Other possible usages of page object include adding headers and footers to PDF documents, adding an image logo to each page, or generating a template PDF on demand.

Example:

3.24.1 How to create a text object in a PDF page

```
using foxit.common;
using foxit.pdf;
using foxit.pdf.graphics;
using foxit.common.fxcrt;
...

// Assuming PDFPage page has been loaded and parsed.

long position = page.GetLastGraphicsObjectPosition(GraphicsObject.Type.e_TypeText);
TextObject text_object = TextObject.Create();
text_object.SetFillColor(0xFFFF7F00);

// Prepare text state.
TextState state = new TextState();
Font font = new Font("Simsun", (int)Font.Styles.e_StylesSmallCap,
Font.Charset.e_CharsetGB2312, 0);
state.font_size = 80.0f;
state.font = font;
state.textmode = TextState.Mode.e_ModeFill;
text_object.SetTextState(page, state, false, 750);

// Set text.
text_object.SetText("Foxit Software");
long last_position = page.InsertGraphicsObject(position, text_object);
...
```

3.24.2 How to add an image logo to a PDF page

```
using foxit.common;
using foxit.pdf;
using foxit.pdf.graphics;
using foxit.common.fxcrt;
...

// Assuming PDFPage page has been loaded and parsed.

long position = page.GetLastGraphicsObjectPosition(GraphicsObject.Type.e_TypeImage);
Image image = new Image(image_file);
ImageObject image_object = ImageObject.Create(page.GetDocument());
image_object.SetImage(image, 0);

float width = image.GetWidth();
float height = image.GetHeight();
float page_width = page.GetWidth();
float page_height = page.GetHeight();

image_object.SetMatrix(new Matrix2D(width, 0, 0, height, (page_width - width) / 2.0f,
(page_height - height) / 2.0f));
page.InsertGraphicsObject(position, image_object);
page.GenerateContent();
```

```
...
```

3.25 Marked content

In PDF document, a portion of content can be marked as marked content element. Marked content helps to organize the logical structure information in a PDF document and enables stylized tagged PDF. Tagged PDF has a standard structure types and attributes that allow page content to be extracted and reused for other purposes. More details about marked content could be found in chapter 10.5 of PDF reference 1.7 [1].

Example:

3.25.1 How to get marked content in a page and get the tag name

```
using foxit;
using foxit.common;
using foxit.common.fxcrt;
using foxit.pdf;
using foxit.pdf.graphics;
using foxit.pdf.objects;

...
// Assuming PDFPage page has been loaded and parsed.

long position = page.GetFirstGraphicsObjectPosition(GraphicsObject.Type.e_TypeText);
GraphicsObject text_obj = page.GetGraphicsObject(position);
MarkedContent content = text_obj.GetMarkedContent();

int nCount = content.GetItemCount();
// Get marked content property
for (int i = 0; i < nCount; i++)
{
     String tag_name = content.GetItemTagName(i);
     int mcid = content.GetItemMCID(i);
}
...
```

3.26 Layer

PDF Layers, in other words, Optional Content Groups (OCG), are supported in Foxit PDF SDK. Users can selectively view or hide the contents in different layers of a multi-layer PDF document. Multi-layers are widely used in many application domains such as CAD drawings, maps, layered artwork and multi-language document, etc.

In Foxit PDF SDK, a PDF layer is associated with a layer node. To retrieve a layer node, user should construct a PDF LayerTree object first and then call function LayerTree.GetRootNode to get the root layer node of the whole layer tree. Furthermore, you can enumerate all the nodes in the layer tree from the root layer node. Foxit PDF SDK provides APIs to get/set layer data, view or hide the contents in different layers, set layers' name, add or remove layers, and edit layers.

Example:

3.26.1 How to create a PDF layer

```
using foxit;
using foxit.common;
using foxit.common.fxcrt;
using foxit.pdf;
 ...
// Assuming PDFDoc doc has been loaded.

LayerTree layertree = new LayerTree(doc);
LayerNode root = layertree.GetRootNode();
...
```

3.26.2 How to set all the layer nodes information

```
using foxit;
using foxit.common;
using foxit.common.fxcrt;
using foxit.pdf;
...
// Assuming PDFDoc doc has been loaded.

public static void SetAllLayerNodesInformation(LayerNode layer_node)
{
     if (layer_node.HasLayer())
     {
        layer_node.SetDefaultVisible(true);
        layer_node.SetExportUsage(LayerTree.UsageState.e_StateUndefined);
        layer_node.SetViewUsage(LayerTree.UsageState.e_StateOFF);
        LayerPrintData print_data = new LayerPrintData("subtype_print",
LayerTree.UsageState.e_StateON);
        layer_node.SetPrintUsage (print_data);
        LayerZoomData zoom_data = new LayerZoomData(1, 10);
        layer_node.SetZoomUsage(zoom_data);
        string new_name = "[View_OFF_Print_ON_Export_Undefined]" + layer_node.GetName();
        layer_node.SetName(new_name);
     }
     int count = layer_node.GetChildrenCount();
     for (int i = 0; i < count; i++)
     {
        using (LayerNode child = layer_node.GetChild(i))
        {
              SetAllLayerNodesInformation(child);
        }
     }
}

LayerTree layertree = new LayerTree(doc);
LayerNode root = layertree.GetRootNode();
if (root.IsEmpty())
{
    return;
}
SetAllLayerNodesInformation(root);
...
```

3.26.3 How to edit layer tree

```
using foxit;
using foxit.common;
using foxit.common.fxcrt;
using foxit.pdf;
 ...
// Assuming PDFDoc doc has been loaded.

LayerTree layertree = new LayerTree(doc);
LayerNode root = layertree.GetRootNode();
int children_count = root.GetChildrenCount();
root.RemoveChild(children_count - 1);
LayerNode child = root.GetChild(children_count - 2);
LayerNode child0 = root.GetChild(0);
child.MoveTo(child0, 0);
child.AddChild(0, "AddedLayerNode", true);
child.AddChild(0, "AddedNode", false);
```

3.27 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**: Foxit PDF SDK provides default Signature handler which supports the following two types of signature filter and subfilter:*

(1) filter: Adobe.PPKLite	*subfilter: adbe.pkcs7.detached*
(2) filter: Adobe.PPKLite	*subfilter: adbe.pkcs7.sha1*

If you use one of the above signature filter and subfilter, you can sign a PDF document and verify the validity of signature by default without needing to register a custom callback.

Example:

3.27.1 How to sigh the PDF document with a signature

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
using foxit.common;
using foxit.pdf;
using foxit;
using foxit.pdf.annots;
using foxit.common.fxcrt;
using System.Runtime.InteropServices;
using foxit.pdf.interform;

static foxit.common.DateTime GetLocalDateTime()
{
    System.DateTimeOffset rime = System.DateTimeOffset.Now;
    foxit.common.DateTime datetime = new foxit.common.DateTime();
    datetime.year = (UInt16)rime.Year;
    datetime.month = (UInt16)rime.Month;
    datetime.day = (ushort)rime.Day;
    datetime.hour = (UInt16)rime.Hour;
    datetime.minute = (UInt16)rime.Minute;
    datetime.second = (UInt16)rime.Second;
    datetime.utc_hour_offset = (short)rime.Offset.Hours;
    datetime.utc_minute_offset = (ushort)rime.Offset.Minutes;
    return datetime;
}

static Signature AddSiganture(PDFPage pdf_page, string sub_filter) {
    float page_height = pdf_page.GetHeight();
    float page_width = pdf_page.GetWidth();
    RectF new_sig_rect = new RectF(0, (float)(page_height*0.9), (float)(page_width*0.4),
page_height);
    // Add a new signature to page.
    Signature new_sig = pdf_page.AddSignature(new_sig_rect);
    if (new_sig.IsEmpty()) return null;
    // Set values for the new signature.
    new_sig.SetKeyValue(Signature.KeyName.e_KeyNameSigner, "Foxit PDF SDK");
    String new_value = String.Format("As a sample for subfilter \"{0}\"", sub_filter);
    new_sig.SetKeyValue(Signature.KeyName.e_KeyNameReason, new_value);
    new_sig.SetKeyValue(Signature.KeyName.e_KeyNameContactInfo, "support@foxitsoftware.com");
    new_sig.SetKeyValue(Signature.KeyName.e_KeyNameDN, "CN=CN,MAIL=MAIL@MAIL.COM");
    new_sig.SetKeyValue(Signature.KeyName.e_KeyNameLocation, "Fuzhou, China");
    new_value = String.Format("As a sample for subfilter \"{0}\"", sub_filter);
    new_sig.SetKeyValue(Signature.KeyName.e_KeyNameText, new_value);
    foxit.common.DateTime sign_time = GetLocalDateTime();
    new_sig.SetSignTime(sign_time);
    String image_file_path = input_path + "FoxitLogo.jpg";
    using (Image image = new Image(image_file_path))
    {
        new_sig.SetImage(image, 0);
        // Set appearance flags to decide which content would be used in appearance.
        int ap_flags = Convert.ToInt32(Signature.APFlags.e_APFlagLabel |
Signature.APFlags.e_APFlagSigner |
                Signature.APFlags.e_APFlagReason | Signature.APFlags.e_APFlagDN |
                Signature.APFlags.e_APFlagLocation | Signature.APFlags.e_APFlagText |
                Signature.APFlags.e_APFlagSigningTime | Signature.APFlags.e_APFlagBitmap);
        new_sig.SetAppearanceFlags(ap_flags);
    }

    return new_sig;
}

static void AdobePPKLiteSignature(PDFDoc pdf_doc) {
    string filter = "Adobe.PPKLite";
    string sub_filter = "adbe.pkcs7.detached";

    using (PDFPage pdf_page = pdf_doc.GetPage(0))
    {
        // Add a new signature to first page.
        using (Signature new_signature = AddSiganture(pdf_page, sub_filter))
        {
            // Set filter and subfilter for the new signature.
            new_signature.SetFilter(filter);
            new_signature.SetSubFilter(sub_filter);

            // Sign the new signature.
            String signed_pdf_path = output_directory + "signed_newsignature.pdf";

            String cert_file_path = input_path + "foxit_all.pfx";
            byte[] cert_file_password = Encoding.ASCII.GetBytes("123456");
            new_signature.StartSign(cert_file_path, cert_file_password,
                            Signature.DigestAlgorithm.e_DigestSHA1, signed_pdf_path,
IntPtr.Zero, null);
            Console.WriteLine("[Sign] Finished!");
        }
    }
}

static void Main(String[] args)
{
    ...
    AdobePPKLiteSignature(pdf_doc);
    ...
}
```

3.28 Long term validation (LTV)

From version 7.0, Foxit PDF SDK provides APIs to establish long term validation (LTV) of signatures, which is mainly used to solve the verification problem of signatures that have already expired. LTV requires DSS (Document Security Store) which contains the verification information of the signatures, as well as DTS (Document Timestamp Signature) which belongs to the type of time stamp signature.

In order to support LTV, Foxit PDF SDK provides:

- Support for adding the signatures of time stamp type, and provides a default signature callback for the subfilter "ETSI.RFC3161".
- TimeStampServerMgr and TimeStampServer classes, which are used to set and manager the server for time stamp. The default signature callback for the subfilter "ETSI.RFC3161" will use the default time stamp server.
- LTVVerifier class which offers the functionalities of verifying signatures and adding DSS information to documents. It also provides a basic default RevocationCallback which is required by LTVVerifier.

Following lists an example about how to establish long term validation of signatures using the default signature callback for subfilter "ETSI.RFC3161" and the default RevocationCallback. For more details, please refer to the simple demo "**ltv**" in the "\examples\simple_demo" folder of the download package.

Example:

3.28.1 How to establish long term validation of signatures using the default signature callback for subfilter "ETSI.RFC3161" and the default RevocationCallback

```
using System;

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

// Initialize time stamp server manager, add and set a default time stamp server, which will
be used by default signature callback for time stamp signature.
TimeStampServerMgr.Initialize();
using (TimeStampServer timestamp_server = TimeStampServerMgr.AddServer(server_name,
server_url, server_username, server_password))
{
  TimeStampServerMgr.SetDefaultServer(timestamp_server);
  // Assume that "signed_pdf_path" represents a signed PDF document which contains signed
signature.
  using (PDFDoc pdf_doc = new PDFDoc(signed_pdf_path))
  {
    pdf_doc.StartLoad(null, true, null);
    // Use LTVVerifier to verify and add DSS.
    using (LTVVerifier ltv_verifier = new LTVVerifier(pdf_doc, true, false, false,
LTVVerifier.TimeType.e_SignatureTSTTime))
    {
      // Set verifying mode which is necessary.
      ltv_verifier.SetVerifyMode(LTVVerifier.VerifyMode.e_VerifyModeETSI);
      SignatureVerifyResultArray sig_verify_result_array = ltv_verifier.Verify();
      for (uint i = 0; i < sig_verify_result_array.GetSize(); i++)
      {
        // ltv state would be e_LTVStateNotEnable here.
        SignatureVerifyResult.LTVState ltv_state =
sig_verify_result_array.GetAt(i).GetLTVState();
        if ((sig_verify_result_array.GetAt(i).GetSignatureState() &
Convert.ToUInt32(Signature.States.e_StateVerifyValid)) > 0)
          ltv_verifier.AddDSS(sig_verify_result_array.GetAt(i));
      }
      // Add a time stamp signature as DTS and sign it. "saved_ltv_pdf_path" represents the
newly saved signed PDF file.
      using (PDFPage pdf_page = pdf_doc.GetPage(0))
      {
        // The new time stamp signature will have default filter name "Adobe.PPKLite" and
default subfilter name "ETSI.RFC3161".
        Signature timestamp_signature = pdf_page.AddSignature(new RectF(), "",
Signature.SignatureType.e_SignatureTypeTimeStamp);
        byte[] empty_byte = Encoding.ASCII.GetBytes("");
        Progressive sign_progressive = timestamp_signature.StartSign("", empty_byte,
Signature.DigestAlgorithm.e_DigestSHA256,
          saved_ltv_pdf_path, IntPtr.Zero, null);
        if (sign_progressive.GetRateOfProgress() != 100)
          sign_progressive.Continue();

        // Then use LTVVeirfier to verify the new signed PDF file.
        using (PDFDoc check_pdf_doc = new PDFDoc(saved_ltv_pdf_path))
        {
          check_pdf_doc.StartLoad(null, true, null);
          // Use LTVVeirfier to verify.
          using (LTVVerifier ltv_verifier = new LTVVerifier(pdf_doc, true, false, false,
LTVVerifier.TimeType.e_SignatureTSTTime))
```

```
          {
            // Set verifying mode which is necessary.
            ltv_verifier.SetVerifyMode(LTVVerifier.VerifyMode.e_VerifyModeETSI);
            SignatureVerifyResultArray sig_verify_result_array = ltv_verifier.Verify();
            for (uint i = 0; i < sig_verify_result_array.GetSize(); i++)
            {
              // ltv state would be e_LTVStateEnable here.
              SignatureVerifyResult.LTVState ltv_state = 
sig_verify_result_array.GetAt(i).GetLTVState();
              ... // User can get other information from SignatureVerifyResult.
            }
          }
        }
      }
    }
  }
}
// Release time stamp server manager when everything is done.
TimeStampServerMgr.Release();
```

3.29 PAdES

From version 7.0, Foxit PDF SDK also supports PAdES (PDF Advanced Electronic Signature) which is the application for CAdES signature in the field of PDF. CAdES is a new standard for advanced digital signature, its default subfilter is "ETSI.CAdES.detached". PAdES signature includes four levels: B-B, B-T, B-LT, and B-LTA.

- B-B: Must include the basic attributes.
- B-T: Must include document time stamp or signature time stamp to provide trusted time for existing signatures, based on B-B.
- B-LT: Must include DSS/VRI to provide certificates and revocation information, based on B-T.
- B-LTA: Must include the trusted time DTS for existing revocation information, based on B-LT.

Foxit PDF SDK provides a default signature callback for the subfilter "ETSI.CAdES.detached" to sign and verify the signatures (with subfilter "ETSI.CAdES.detached"). It also provides TimeStampServerMgr and TimeStampServer classes to set and manager the server for time stamp. The default signature callback for the subfilter "ETSI.CAdES.detached" will use the default time stamp server.

Foxit PDF SDK provides functions to get the level of PAdES from signature, and application level can also judge and determine the level of PAdES according to the requirements of each level. For more details about how to add, sign and verify a PAdES signature in PDF document, please refer to the simple demo "**pades**" in the "\examples\simple_demo" folder of the download package.

3.30 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:

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

```
using foxit.common;
using foxit.pdf;
using foxit;
using foxit.pdf.annots;
using foxit.common.fxcrt;
using foxit.pdf.annots;
using foxit.pdf.actions;

// Add link annotation
Link link = null;
Annot annot = null;
using (annot = page.AddAnnot(Annot.Type.e_Link, new RectF(350, 350, 380, 400)))
using (link = new Link(annot))
using (PDFDoc doc = page.GetDocument())
{
    foxit.pdf.actions.Action action = null;
    link.SetHighlightingMode(Annot.HighlightingMode.e_HighlightingToggle);

    // Add action for link annotation
    URIAction uriaction = null;
    using (action = foxit.pdf.actions.Action.Create(doc,
foxit.pdf.actions.Action.Type.e_TypeGoto))
    using (uriaction = new URIAction(action))
    {
        uriaction.SetTrackPositionFlag(true);
        uriaction.SetURI("www.foxitsoftware.com");
        link.SetAction(uriaction);

        // Appearance should be reset.
        link.ResetAppearanceStream();
    }
}
```

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

```
using foxit.common;
using foxit.pdf;
using foxit;
using foxit.pdf.annots;
using foxit.common.fxcrt;
using foxit.pdf.annots;
using foxit.pdf.actions;

using foxit.common;
using foxit.pdf;
using foxit;
using foxit.pdf.annots;
using foxit.common.fxcrt;
using foxit.pdf.annots;
using foxit.pdf.actions;

Link link = null;
Annot annot = null;
using (annot = page.AddAnnot(Annot.Type.e_Link, new RectF(350, 350, 380, 400)))
using (link = new Link(annot))
using (PDFDoc doc = page.GetDocument())
{
    foxit.pdf.actions.Action action = null;
    link.SetHighlightingMode(Annot.HighlightingMode.e_HighlightingToggle);

    // Add action for link annotation
    GotoAction gotoaction = null;
    using (action = foxit.pdf.actions.Action.Create(doc,
foxit.pdf.actions.Action.Type.e_TypeGoto))
    using (gotoaction = new GotoAction(action))
    {
        Destination dest = Destination.CreateXYZ(doc, 0, 0, 0, 0);
        gotoaction.SetDestination(dest);
        link.SetAction(gotoaction);
        // Appearance should be reset.
        link.ResetAppearanceStream();
    }
}
```

3.31 JavaScript

JavaScript was created to offload Web page processing from a server onto a client in Web-based applications. Foxit PDF SDK JavaScript implements extensions, in the form of new objects and their accompanying methods and properties, to the JavaScript language. It enables a developer to manage document security, communicate with a database, handle file attachments, and manipulate a PDF file so that it behaves as an interactive, web-enabled form, and so on.

JavaScript action is an action that causes a script to be compiled and executed by the JavaScript interpreter. Class foxit.pdf.actions.JavaScriptAction is derived from Action and offers functions to get/set JavaScript action data.

Example:

3.31.1 How to add JavaScript Action to Document

```
using foxit.pdf.annots;
using foxit.pdf.actions;

// Load Document doc
...
using (JavaScriptAction javascipt_action = new JavaScriptAction(action))
{
   javascipt_action.SetScript("app.alert(\"Hello Foxit \");");
   using (AdditionalAction aa = new AdditionalAction(doc))
   {
      aa.SetAction(AdditionalAction.TriggerEvent.e_TriggerDocWillClose, javascipt_action);
```

```
    }
}
...
```

3.31.2 How to add JavaScript Action to Annotation

```
using foxit.pdf.annots;
using foxit.pdf.actions;
...

// Load Document and get a widget annotation.
...
using (JavaScriptAction javascipt_action = new JavaScriptAction(action))
{
   javascipt_action.SetScript("app.alert(\"Hello Foxit \");");
   using (AdditionalAction aa = new AdditionalAction(annot))
   {
      aa.SetAction(AdditionalAction.TriggerEvent.e_TriggerAnnotMouseButtonPressed,
javascipt_action);
    }
}
...
```

3.31.3 How to add JavaScript Action to FormField

```
using foxit.pdf.interform;
using foxit.pdf.actions;
...

// Load Document and get a form field.
...

using (JavaScriptAction javascipt_action = new JavaScriptAction(action))
{
   javascipt_action.SetScript("AFSimple_Calculate(\"SUM\", new Array (\"Text Field0\", \"Text
Field1\"));");
   using (AdditionalAction aa = new AdditionalAction(field))
   {
      aa.SetAction(AdditionalAction.TriggerEvent.e_TriggerFieldRecalculateValue,
javascipt_action);
    }
}
...
```

3.32 Redaction

Redaction is the process of removing sensitive information while keeping the document's layout. It allows users to permanently remove (redact) visible text and images from PDF documents to protect confidential information, such as social security numbers, credit card information, product release dates, and so on.

Redaction is a type of markup annotation, which is used to mark some contents of a PDF file and then the contents will be removed once the redact annotations are applied.

To do Redaction, you can use the following APIs:

- Call function foxit.addon.Redaction.Redaction to create a redaction module. If module "Redaction" is not defined in the license information which is used in function common.Library.Initialize, it means user has no right in using redaction related functions and this constructor will throw exception foxit.common.ErrorCode.e_ErrInvalidLicense .
- Then call function foxit.addon.Redaction.MarkRedactAnnot to create a redaction object and mark page contents (text object, image object, and path object) which are to be redacted.
- Finally call function foxit.addon.Redaction.Apply to apply redaction in marked areas: remove the text or graphics under marked areas permanently.

Note*: To use the redaction feature, please make sure the license key has the permission of the 'Redaction' module*

Example:

3.32.1 How to redact the text "PDF" on the first page of a PDF

```
using foxit;
using foxit.common;
using foxit.common.fxcrt;
using foxit.pdf;
using foxit.addon;
using foxit.pdf.annots;

...
using (Redaction redaction = new Redaction(doc))
{
    using (PDFPage page = doc.GetPage(0))
    {
        // Parse PDF page.
        page.StartParse((int)foxit.pdf.PDFPage.ParseFlags.e_ParsePageNormal, null, false);
        TextPage text_page = new TextPage(page,
(int)foxit.pdf.TextPage.TextParseFlags.e_ParseTextNormal);
        TextSearch text_search = new TextSearch(text_page);
        text_search.SetPattern("PDF");
        RectFArray rect_array = new RectFArray();
        while (text_search.FindNext())
        {
            RectFArray itemArray = text_search.GetMatchRects();
            rect_array.InsertAt(rect_array.GetSize(), itemArray);
        }
        if (rect_array.GetSize() > 0)
        {
            Redact redact = redaction.MarkRedactAnnot(page, rect_array);
            redact.ResetAppearanceStream();
            doc.SaveAs(output_path + "AboutFoxit_redected_default.pdf",
(int)foxit.pdf.PDFDoc.SaveFlags.e_SaveFlagNormal);

            // set border color to Green
            redact.SetBorderColor(0x00FF00);
            // set fill color to Blue
            redact.SetFillColor(0x0000FF);
            // set rollover fill color to Red
            redact.SetApplyFillColor(0xFF0000);
            redact.ResetAppearanceStream();
            doc.SaveAs(output_path + "AboutFoxit_redected_setColor.pdf",
(int)foxit.pdf.PDFDoc.SaveFlags.e_SaveFlagNormal);

            redact.SetOpacity((float)0.5);
            redact.ResetAppearanceStream();
            doc.SaveAs(output_path + "AboutFoxit_redected_setOpacity.pdf",
(int)foxit.pdf.PDFDoc.SaveFlags.e_SaveFlagNormal);

            if (redaction.Apply())
                Console.WriteLine("Redact page(0) succeed.");
            else
                Console.WriteLine("Redact page(0) failed.");
            redact.Dispose();
        }
    }
}
```

3.33 Comparison

Comparison feature lets you see the differences in two versions of a PDF. Foxit PDF SDK provides APIs to compare two PDF documents page by page, the differences between the two documents will be returned. For now, it only supports comparing the text of the PDF document, and in the next release, more PDF elements will be supported.

The differences can be defined into three types: delete, insert and replace. You can save these differences into a PDF file and mark them as annotations.

***Note**: To use the comparison feature, please make sure the license key has the permission of the 'Comparison' module*

Example:

3.33.1 How to compare two PDF documents and save the differences between them into a PDF file

```
using foxit;
using foxit.common;
using foxit.common.fxcrt;
using foxit.pdf;
using foxit.pdf.annots;
using foxit.addon;
...

using (PDFDoc base_doc = new PDFDoc("input_base_file"))
{
    error_code = base_doc.Load(null);
    if (error_code != ErrorCode.e_ErrSuccess)
    {
        Library.Release();
        return;
    }

    using (PDFDoc compared_doc = new PDFDoc("input_compared_file"))
    {
        error_code = compared_doc.Load(null);
        if (error_code != ErrorCode.e_ErrSuccess)
        {
            Library.Release();
            return;
        }

        using (Comparison comparison = new Comparison(base_doc, compared_doc))
        {
            // Start comparing.
            CompareResults result = comparison.DoCompare(0, 0,
(int)Comparison.CompareType.e_CompareTypeText);
            CompareResultInfoArray oldInfo = result.results_base_doc;
            CompareResultInfoArray newInfo = result.results_compared_doc;
            uint oldInfoSize = oldInfo.GetSize();
            uint newInfoSize = newInfo.GetSize();

            using (PDFPage page = compared_doc.GetPage(0))
            {
                 for (uint i = 0; i < newInfoSize; i++)
                 {
                     CompareResultInfo item = newInfo.GetAt(i);
                     CompareResultInfo.CompareResultType type = item.type;
                     if (type ==
CompareResultInfo.CompareResultType.e_CompareResultTypeDeleteText)
                     {
                         String res_string = String.Format("\"{0}\"", item.diff_contents);

                         // Add stamp to mark the "delete" type differences between the two
documents.
                         CreateDeleteTextStamp(page, item.rect_array, 0xff0000, res_string,
"Compare : Delete", "Text");
                     }
                     else if (type ==
CompareResultInfo.CompareResultType.e_CompareResultTypeInsertText)
                     {
                         String res_string = String.Format("\"{0}\"", item.diff_contents);

                         // Highlight the "insert" type differences between the two documents.
                         CreateHighlightRect(page, item.rect_array, 0x0000ff, res_string,
"Compare : Insert", "Text");
                     }
                     else if (type ==
CompareResultInfo.CompareResultType.e_CompareResultTypeReplaceText)
                     {
                         String res_string = String.Format("[Old]: \"{0}\"\r\n[New]: \"{1}\"",
oldInfo.GetAt(i).diff_contents, item.diff_contents);

                         // Highlight the "replace" type differences between the two
documents.
                         CreateHighlightRect(page, item.rect_array, 0xe7651a, res_string,
"Compare : Replace", "Text");
                     }
                 }
            }

            // Save the comparison result to a PDF file.
            compared_doc.SaveAs(output_path + "result.pdf",
(int)PDFDoc.SaveFlags.e_SaveFlagNormal);
        }
    }
}
```

*Note: for `CreateDeleteTextStamp` and `CreateHighlightRect` functions, please refer to the simple demo "**pdfcompare**" located in the "\examples\simple_demo" folder of the download package.*

3.34 OCR

Optical Character Recognition, or OCR, is a software process that enables images or printed text to be translated into machine-readable text. OCR is most commonly used when scanning paper documents to create electronic copies, but can also be performed on existing electronic documents (e.g. PDF).

This section will provide instructions on how to set up your environment for the OCR feature module using Foxit PDF SDK for .Net.

3.34.1 System requirements

Platform: Windows

Programming Language: C++, Java, C#

License Key requirement: 'OCR' module permission in the license key

SDK Version: Foxit PDF SDK for Windows (C++, Java, C#) 6.4 or higher

3.34.2 Trial Limit for SDK OCR Add-on Module

For the trial version, there are three trail limits that you should notice:

1. Allow 30 consecutive natural days to evaluate SDK from the first time of OCREngine initialization.
2. Allow up to 5000 pages to be converted using OCR from the first time of OCREngine initialization.
3. Trail watermarks will be generated on the PDF pages. This limit is used for all of the SDK modules.

3.34.3 OCR resource files

Please contact Foxit support team or sales team to get the OCR resource files package.

After getting the package, extract it to a desired directory (for example, extract the package to a directory named "ocr_addon"), and then you can see the resource files for OCR are as follows:

- **debugging_files:** Resource files used for debugging the OCR project. These file(s) cannot be distributed.

- **language_resource_CJK:** Resource files for CJK language, including: Chinese-Simplified, Chinese-Traditional, Japanese, and Korean.
- **language_resources_noCJK:** Resource files for the languages except CJK, including: Basque, Bulgarian, Catalan, Croatian, Czech, Danish, Dutch, English , Estonian, Faeroese, Finnish, French, Galician, German, Greek, Hebrew, Hungarian, Icelandic, Italian, Latvian(Lettish), Lithuanian, Macedonian, Maltese, Norwegian, Polish, Portuguese, Romanian, Russian, Serbian, Slovak, Slovenian, Spanish, Swedish, Thai, Turkish, Ukrainian.
- **win32_lib:** 32-bit library resource files
- **win64_lib:** 64-bit library resource files
- **readme.txt:** A txt file for introducing the role of each folder in this directory, as well as how to use those resource files for OCR.

3.34.4 How to run the OCR demo

Foxit PDF SDK for .Net provides an OCR demo located in the "\examples\simple_demo\ocr" folder to show you how to use Foxit PDF SDK to do OCR for a PDF page or a PDF document.

3.34.4.1 Load the OCR demo in Visual Studio

To load the OCR demo in your specific environment, please choose one of the following ways:

1) Load the visual studio solution files "simple_demo_vs2010.sln" or "simple_demo_vs2015.sln" or "simple_demo_vs2017.sln" (depending on your Visual Studio version) in the "\examples\simple_demo" folder. Right-click the ocr demo, choose **Set as StartUp Project**.
2) Load the "ocr_vs2010.csproj" or "ocr_vs2015.csproj" or "ocr_vs2017.csproj" (depending on your Visual Studio version) in the "\examples\simple_demo\ocr" folder.

3.34.4.2 Build an OCR resource directory

Before running the OCR demo, you should first build an OCR resource directory, and then pass the directory to Foxit PDF SDK API **OCREngine.Initialize** to initialize OCR engine.

To build an OCR resource directory, please follow the steps below:

1) Create a new folder to add the resources. For example, "D:/ocr_resources".
2) Add the appropriate library resource based on the platform architecture.
    - For **win32**, copy **all the files** under "ocr_addon/win32_lib" folder to "D:/ocr_resources".
    - For **win64**, copy **all the files** under "ocr_addon/win64_lib" folder to "D:/ocr_resources".

3) Add the language resource.

- For CJK (Chinese-Simplified, Chinese-Traditional, Japanese, and Korean), copy **all the files** under "ocr_addon/language_resource_CJK" folder to "D:/ocr_resources".
- For all other languages except CJK, copy **all the files** under "ocr_addon/language_resources_noCJK" folder to "D:/ocr_resources".
- For all the supported languages, copy **all the files** under "ocr_addon/language_resource_CJK" and "ocr_addon/language_resources_noCJK" folders to "D:/ocr_resources".

4) (Optional) Add debugging file resource if you need to debug the demo.

- For win32, copy the file(s) under "ocr_addon/debugging_files/win32" folder to "D:/ocr_resources".
- For win64, copy the file(s) under "ocr_addon/debugging_files/win64" folder to "D:/ocr_resources".

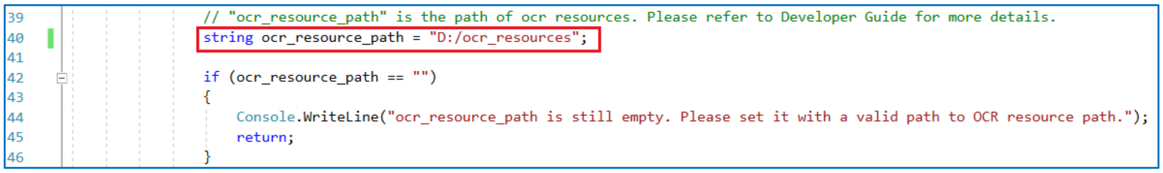
Note*: The debugging files should be exclusively used for testing purposes. So, you cannot distribute them.*

3.34.4.3 Configure the demo

After building the OCR resource directory, configure the demo in the "\examples\simple_demo\ocr\ocr.cs" file.

Specify the OCR resource directory

Add the OCR resource directory as follows, which will be used to initialize the OCR engine.

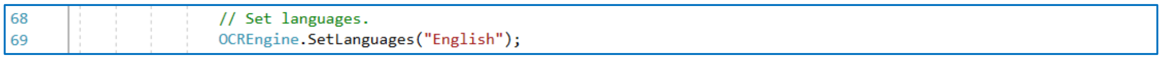


```
// "ocr_resource_path" is the path of ocr resources. Please refer to Developer Guide for more details.
string ocr_resource_path = "D:/ocr_resources";

if (ocr_resource_path == "")
{
    Console.WriteLine("ocr_resource_path is still empty. Please set it with a valid path to OCR resource path.");
    return;
}
```

Choose the language resource

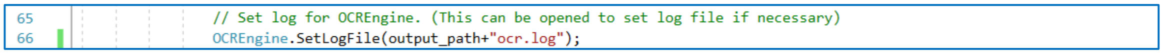
You will need to set the language used by the OCR engine into the demo code. This is done with the **OCREngine.SetLanguages** method and is set to "English" by default.



```
// Set languages.
OCREngine.SetLanguages("English");
```

(Optional) Set log for OCREngine

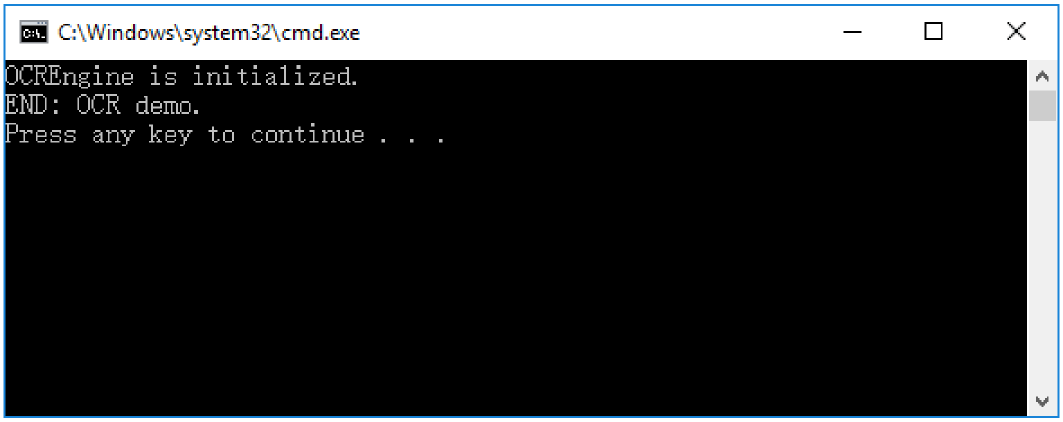
If you add the debugging file resource to the OCR resource directory, and want to print the entire log of the OCR Engine, please uncomment the **OCREngine.SetLogFile** method as below:



```
// Set log for OCREngine. (This can be opened to set log file if necessary)
OCREngine.SetLogFile(output_path+"ocr.log");
```

3.34.4.4 *Run the demo*

Once you run the demo successfully, the console will print the following by default:



```
OCREngine is initialized.
END: OCR demo.
Press any key to continue . . .
```

The demo will OCR the default document ("\examples\simple_demo\input_files\ocr\AboutFoxit_ocr.pdf") in four different ways, which will output four different PDFs in the output folder ("\examples\simple_demo\output_files\ocr"):

- OCR Editable PDF - ocr_doc_editable.pdf
- OCR Searchable PDF - ocr_doc_searchable.pdf
- OCR Editable PDF Page - ocr_page_editable.pdf
- OCR Searchable PDF Page - ocr_page_searchable.pdf

3.35 Compliance

PDF Compliance

Foxit PDF SDK supports to convert PDF versions among PDF 1.3, PDF 1.4, PDF 1.5, PDF 1.6 and PDF 1.7. When converting to PDF 1.3, if the source document contains transparency data, then it will be converted to PDF 1.4 instead of PDF 1.3 (PDF 1.3 does not support transparency). If the source document does not contain any transparency data, then it will be converted to PDF 1.3 as expected.

PDF/A Compliance

PDF/A is an ISO-standardized version of the PDF specialized for use in the archiving and long-term preservation of electronic documents. PDF/A differs from PDF by prohibiting features unsuitable for long-term archiving, such as font linking (as opposed to font embedding), encryption, JavaScript, audio, video and so on.

Foxit PDF SDK provides APIs to verify whether a PDF is compliance with PDF/A standard, or convert a PDF to be compliance with PDF/A standard. It supports the PDF/A version including PDF/A-1a, PDF/A-1b, PDF/A-2a, PDF/A-2b, PDF/A-2u, PDF/A-3a, PDF/A-3b, PDF/A-3u (ISO 19005- 1, 19005 -2 and 19005-3).

This section will provide instructions on how to set up your environment for running the 'compliance' demo.

3.35.1 System Requirements

Platform: Windows, Linux, Mac

Programming Language: C++, Java, C#, Objective-C

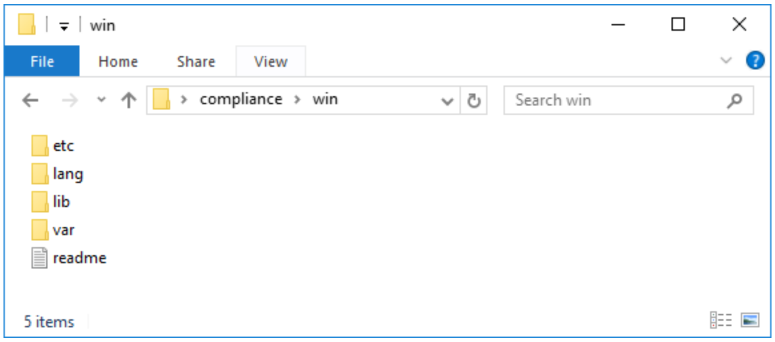
License Key requirement: 'Compliance' module permission in the license key

SDK Version: Foxit PDF SDK 6.4 or higher (for PDF Compliance, it requires Foxit PDF SDK 7.1 or higher)

3.35.2 Compliance resource files

Please contact Foxit support team or sales team to get the Compliance (PDF-A) resource files package.

After getting the package, extract it to a desired directory (for example, extract the package to a directory: "**compliance/win**"), and then you can see the resource files for Compliance are as follows:



3.35.3 How to run the compliance demo

Foxit PDF SDK provides a **compliance** demo located in the "\examples\simple_demo\compliance" folder to show you how to use Foxit PDF SDK to verify whether a PDF is compliance with PDF/A standard, and convert a PDF to be compliance with PDF/A standard, as well as convert PDF versions.

3.35.3.1 Load the compliance demo in Visual Studio

To load the **compliance** demo in your specific environment, please choose one of the following ways:

1) Load the visual studio solution files "simple_demo_vs2010.sln" or "simple_demo_vs2015.sln" or "simple_demo_vs2017.sln" (depending on your Visual Studio version) in the "\examples\simple_demo" folder. Right-click the compliance demo, choose **Set as StartUp Project**.
2) Load the "compliance_vs2010.csproj" or "ocr_vs2015.csproj" or "ocr_vs2017.csproj" (depending on your Visual Studio version) in the "\examples\simple_demo\compliance" folder.

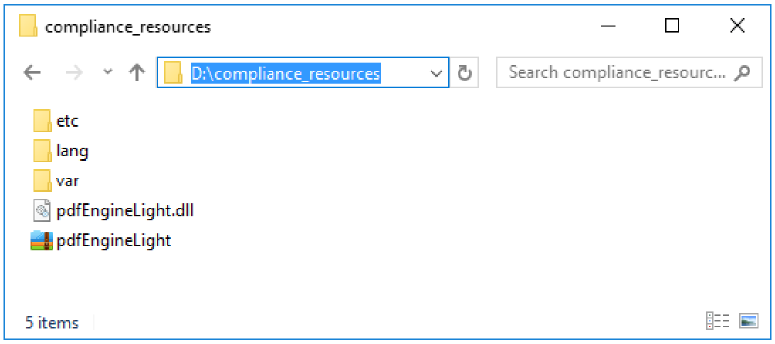
3.35.3.2 Build a compliance resource directory

Before running the **compliance** demo, you should first build a compliance resource directory, and then pass the directory to Foxit PDF SDK API **ComplianceEngine.Initialize** to initialize compliance engine.

To build a compliance resource directory, please follow the steps below:

1) Create a new folder to add the resources. For example, "D:/compliance_resources".
2) Copy the whole folders of "**ect**", "**lang**", "**var**" under the "compliance/win" to "D:/compliance_resources".
3) Add the appropriate library resource based on the platform architecture.
    - For **win32**, copy **all the files** under "compliance/win/lib/x86" folder to "D:/compliance_resources".
    - For **win64**, copy **all the files** under "compliance/win/lib/x64" folder to "D:/compliance_resources".

For example, use win32 platform architecture, then the compliance resource directory should be as follows:



D:\compliance_resources

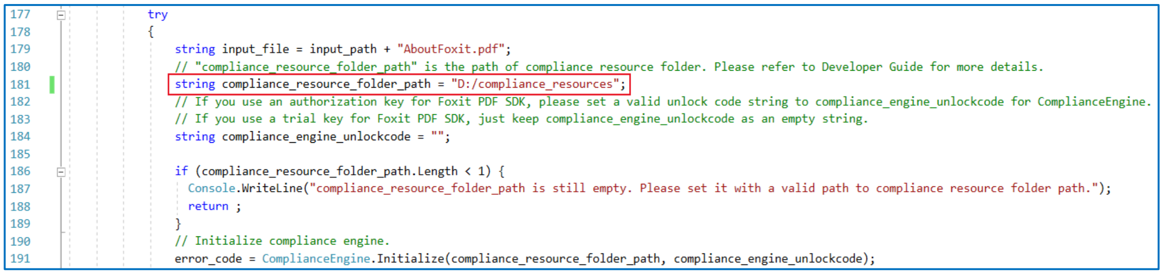
etc
lang
var
pdfEngineLight.dll
pdfEngineLight

3.35.3.3 Configure the demo

After building the compliance resource directory, configure the demo in the "\examples\simple_demo\compliance\compliance.cs" file.

Specify the compliance resource directory

Add the compliance resource directory as follows, which will be used to initialize the compliance engine.



```
try
{
    string input_file = input_path + "AboutFoxit.pdf";
    // "compliance_resource_folder_path" is the path of compliance resource folder. Please refer to Developer Guide for more details.
    string compliance_resource_folder_path = "D:/compliance_resources";
    // If you use an authorization key for Foxit PDF SDK, please set a valid unlock code string to compliance_engine_unlockcode for ComplianceEngine.
    // If you use a trial key for Foxit PDF SDK, just keep compliance_engine_unlockcode as an empty string.
    string compliance_engine_unlockcode = "";

    if (compliance_resource_folder_path.Length < 1) {
      Console.WriteLine("compliance_resource_folder_path is still empty. Please set it with a valid path to compliance resource folder path.");
      return ;
    }
    // Initialize compliance engine.
    error_code = ComplianceEngine.Initialize(compliance_resource_folder_path, compliance_engine_unlockcode);
```

***Note**: If you have purchased an authorization license key (includes 'Compliance' module permission), Foxit sales team will send you an extra unlock code for initializing compliance engine.*

(Optional) Set language for compliance engine

ComplianceEngine.SetLanguage function is used to set language for compliance engine. The default language is "English", and the supported languages are as follows:

"Czech", "Danish", "Dutch", "English", "French", "Finnish", "German", "Italian", "Norwegian", "Polish", "Portuguese", "Spanish", "Swedish", "Chinese-Simplified", "Chinese-Traditional", "Japanese", "Korean".

For example, set the language to "Chinese-Simplified":

```
// Set language. The default language is "English".
ComplianceEngine.SetLanguage("Chinese-Simplified");
```

(Optional) Set a temp folder for compliance engine

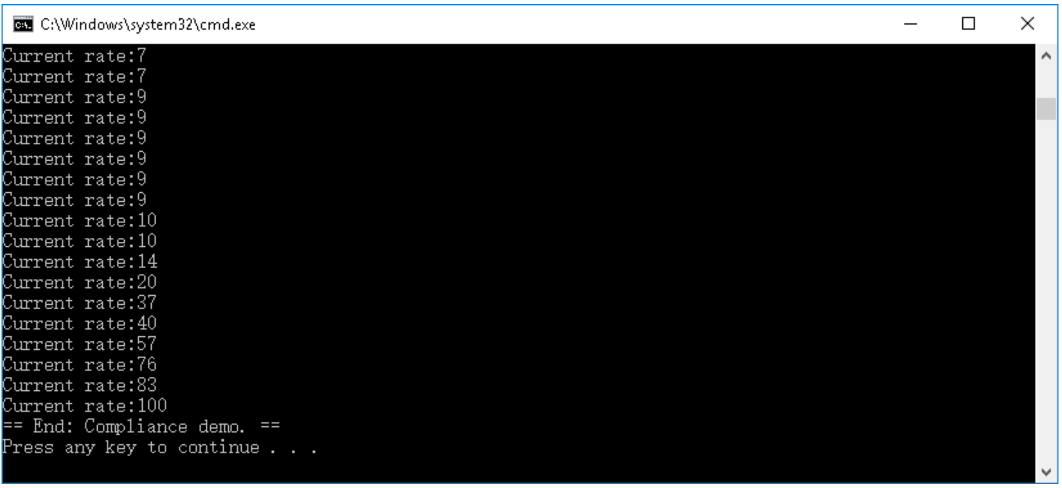
ComplianceEngine.SetTempFolderPath function is used to set a temp folder to store several files for proper processing (e.g verifying or converting). If no custom temp folder is set by this function, the default temp folder in system will be used.

For example, set the path to "D:/compliance_temp" (should be a valid path).

```
// Set custom temp folder path for ComplianceEngine.
ComplianceEngine.SetTempFolderPath(L"D:/compliance_temp");
```

3.35.3.4 Run the demo

Once you run the demo successfully, the console will print the following by default:



```
Current rate:7
Current rate:7
Current rate:9
Current rate:9
Current rate:9
Current rate:9
Current rate:9
Current rate:9
Current rate:10
Current rate:10
Current rate:14
Current rate:20
Current rate:37
Current rate:40
Current rate:57
Current rate:76
Current rate:83
Current rate:100
== End: Compliance demo. ==
Press any key to continue . . .
```

.

The demo will

- verify whether the PDF ("\examples\simple_demo\input_files\AboutFoxit.pdf") is compliance with PDF/A-1a standard, and convert the PDF to be compliance with PDF/A-1a standard.
- convert PDF file ("\examples\simple_demo\input_files\AF_ImageXObject_FormXObject.pdf") to PDF-1.4 and PDF-1.7.

The output files are located in "\examples\simple_demo\output_files\compliance" folder.

3.36 Optimization

Optimization feature can reduce the size of PDF files to save disk space and make files easier to send and store, through compressing images, deleting redundant data, discarding useless user data and so on. From version 7.0, optimization module provides functions to compress the color/grayscale/monochrome images in PDF files to reduce the size of the PDF files.

***Note**: To use the Optimization feature, please make sure the license key has the permission of the 'Optimization' module.*

Example:

3.36.1 How to optimize PDF files by compressing the color/grayscale/monochrome images

```
using System;

using foxit;
using foxit.common;
using foxit.common.fxcrt;
using foxit.pdf;
using foxit.addon;
using foxit.addon.optimization;

using (PDFDoc doc = new PDFDoc("input_pdf_file"))
```

```
{
   error_code = doc.Load(null);
   if (error_code != ErrorCode.e_ErrSuccess)
   {
       Console.WriteLine("Document Load Error: {0}\n", error_code);
       return;
   }
   using (Optimization_Pause pause = new Optimization_Pause(0, true))
   using (OptimizerSettings settings = new OptimizerSettings())
   {
       Console.WriteLine("Optimized Start.\n");
       Progressive progressive = Optimizer.Optimize(doc, settings, pause);
       Progressive.State progress_state = Progressive.State.e_ToBeContinued;
       while (Progressive.State.e_ToBeContinued == progress_state)
       {
            progress_state = progressive.Continue();
            int percent = progressive.GetRateOfProgress();
            Console.WriteLine("Optimize progress percent: {0}%\n", percent);
       }
       if (Progressive.State.e_Finished == progress_state)
       {
            doc.SaveAs("ImageCompression_Optimized.pdf",
(int)PDFDoc.SaveFlags.e_SaveFlagRemoveRedundantObjects);
       }
       Console.WriteLine("Optimized Finish.\n");
    }
}
```

3.37 HTML to PDF Conversion

For some large HTML files or a webpage which contain(s) many contents, it is not convenient to print or archive them directly. Foxit PDF SDK provides APIs to convert the online webpage or local HTML files like invoices or reports into PDF file(s), which makes them easier to print or archive. In the process of conversion from HTML to PDF, Foxit PDF SDK also supports to create and add PDF Tags based on the organizational structure of HTML.

This section will provide instructions on how to set up your environment for running the 'html2pdf' demo.

3.37.1 System requirements

Platform: Windows, Mac

Programming Language: C++, Java, C#, Objective-C

License Key requirement: 'Conversion' module permission in the license key

SDK Version: Foxit PDF SDK 7.0 or higher

3.37.2 HTML to PDF engine files

Please contact Foxit support team or sales team to get the HTML to PDF engine files package.

After getting the package, extract it to a desired directory (for example, extract the package to a directory: "**D:/htmltopdf/win**").

3.37.3 How to run the html2pdf demo

Foxit PDF SDK provides a html2pdf demo located in the "\examples\simple_demo\html2pdf" folder to show you how to use Foxit PDF SDK to convert from html to PDF.

3.37.3.1 Configure the demo

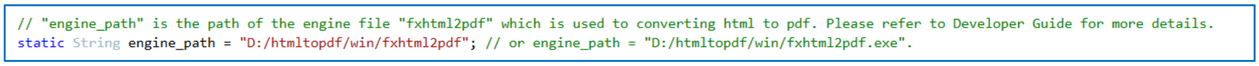
For html2pdf demo, you can configure the demo in the "\examples\simple_demo\html2pdf\html2pdf.cs" file, or you can configure the demo with parameters directly in a command prompt. Following will configure the demo in "html2pdf.cs" file on Windows.

To load the html2pdf demo in Visual Studio, please choose one of the following ways:

1) Load the visual studio solution files "simple_demo_vs2010.sln" or "simple_demo_vs2015.sln" or "simple_demo_vs2017.sln" (depending on your Visual Studio version) in the "\examples\simple_demo" folder. Right-click the html2pdf demo, choose **Set as StartUp Project**.
2) Load the "html2pdf_vs2010.csproj" or "html2pdf_vs2015.csproj" or "html2pdf_vs2017.csproj" (depending on your Visual Studio version) in the "\examples\simple_demo\html2pdf" folder.

Specify the html2pdf engine directory

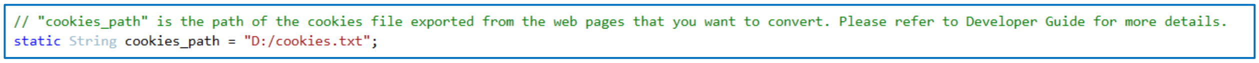
In the "html2pdf.cs" file, add the path of the engine file "fxhtml2pdf.exe" as follows, which will be used to convert html files to PDF files.



```
// "engine_path" is the path of the engine file "fxhtml2pdf" which is used to converting html to pdf. Please refer to Developer Guide for more details.
static String engine_path = "D:/htmltopdf/win/fxhtml2pdf"; // or engine_path = "D:/htmltopdf/win/fxhtml2pdf.exe".
```

(Optional) Specify cookies file path

Add the path of the cookies file exported from the web pages that you want to convert. For example,

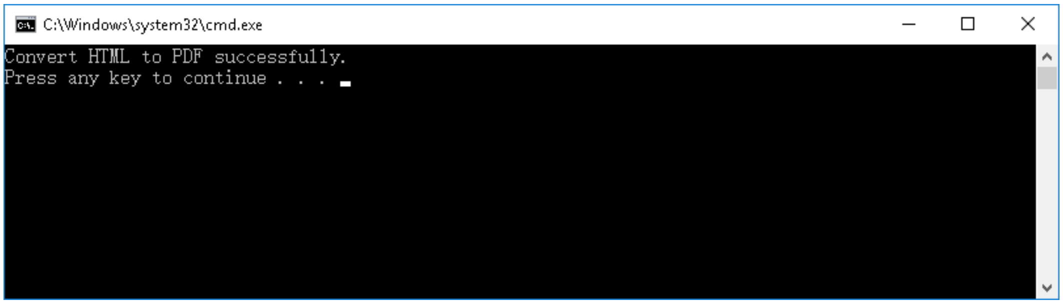


```
// "cookies_path" is the path of the cookies file exported from the web pages that you want to convert. Please refer to Developer Guide for more details.
static String cookies_path = "D:/cookies.txt";
```

3.37.3.2 Run the demo

Run the demo without parameters

Once you run the demo successfully, the console will print the following by default:

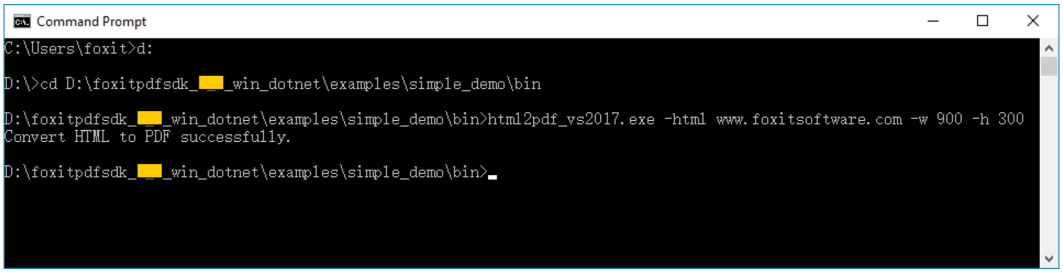


```
Convert HTML to PDF successfully.
Press any key to continue . . .
```

Run the demo with parameters

After building the demo successfully, open a command prompt, navigate to "\examples\simple_demo\bin", type "html2pdf_vs2017.exe --help" for example to see how to use the parameters to execute the program.

For example, convert the URL web page "www.foxitsoftware.com" into a PDF with setting the page width to 900 points and the page height to 300 points:



```
C:\Users\foxit>d:

D:\>cd D:\foxitpdfsdk_[redacted]_win_dotnet\examples\simple_demo\bin

D:\foxitpdfsdk_[redacted]_win_dotnet\examples\simple_demo\bin>html2pdf_vs2017.exe -html www.foxitsoftware.com -w 900 -h 300
Convert HTML to PDF successfully.

D:\foxitpdfsdk_[redacted]_win_dotnet\examples\simple_demo\bin>
```

The output file is located in "\examples\simple_demo\output_files\html2pdf" folder.

Parameters Description

Basic Syntax:

html2pdf *<-html <the url or html path>> <-o <output pdf path>> <-engine <htmltopdf engine path>>*
[-w <page width>] [-h <page height>] [-ml <margin left>] [-mr <margin right>]
[-mt <margin top>] [-mb <margin bottom>] [-r <page rotate degree>] [-mode <page mode>]
[-scale <whether scale page>] [-link <whether convert link>] [-tag <whether generate tag>]
[-cookies <cookies file path>] [-timeout <timeout>]
html2pdf *--help*

Note:

- <> required
- [ ] optional

Parameters	Description
--help	The usage description of the parameters.

-html	The url or html file path. For examples '-html www.foxitsoftware.com'.
-o	The path of the output PDF file.
-engine	The path of the engine file "fxhtml2pdf.exe".
-w	The page width of the output PDF file in points.
-h	The page height of the output PDF file in points.
-r	The page rotate for the output PDF file. • 0 : 0 degree. • 1 : 90 degree. • 2 : 180 degree. • 3 : 270 degree.
-ml	The left margin of the pages for the output PDF file.
-mr	The right margin of the pages for the output PDF file.
-mt	The top margin of the pages for the output PDF file.
-mb	The bottom margin of the pages for the output PDF file.
-mode	The page mode for the output PDF file. • 0 : single page mode. • 1 : multiple pages mode.
-scale	Whether to scale pages. • 'yes' : scale pages. • 'no' : No need to scale pages.
-link	Whether to convert links. • 'yes' : convert links. • 'no' : No need to convert links.
-tag	Whether to generate tag. • 'yes' : generate tag. • 'no' : No need to generate tag.
-cookies	The path of the cookies file exported from a URL that you want to convert.
-timeout	The timeout of loading webpages.

FAQ

1. How to fix the "'xcopy' exited with code 9009" error when building demos in Visual Studio?

   When building demos in Visual Studio, if you encounter the error "'xcopy' exited with code 9009" as follows:

   ```
   'xcopy ..\..\..\..\..\..\..\lib\gsdk_sn.txt ..\..\..\ /y > null
   xcopy ..\..\..\..\..\..\..\lib\gsdk_key.txt ..\..\..\ /y > null
   xcopy ..\..\..\..\..\..\..\lib\$(PlatformName)_vc10\fsdk.dll ..\..\..\ /y > null
   xcopy ..\..\..\..\..\..\..\lib\$(PlatformName)_vc10\fsdk_dotnet.dll ..\..\..\ /y > null'
   exited with code 9009
   ```

   Please check the following points:

   1) Check whether the **xcopy.exe** is in the *"%SystemRoot%\System32"* directory, if not, copy one from another machine.

   2) Check whether the system **PATH** environment variables have been set correctly. It should contain *"%SystemRoot%\System32;%SystemRoot%;"*, if the environment variables for **xcopy** is right, but it still reports the error, please put the path of **xcopy** in front of others. Maybe some other environment variables have spell mistakes, so that cause the subsequent environment variables are invalid. Please check it.

   After checking, open a command prompt, type xcopy command, if it can be recognized, close Visual Studio, and restart the demos. The error should be fixed.

2. **How do I get text objects in a specified position of a PDF and change the contents of the text objects?**

   To get text objects in a specified position of a PDF and change the contents of the text objects using Foxit PDF SDK, you can follow the steps below:

   1) Open a PDF file.

   2) Load PDF pages and get the page objects.

   3) Use **PDFPage.GetGraphicsObjectAtPoint** to get the text object at a certain position. Note: use the page object to get rectangle to see the position of the text object.

   4) Change the contents of the text objects and save the PDF document.

   Following is the sample code:

```
using foxit.common;
using foxit.pdf;
using foxit.pdf.graphics;
```

```
using foxit.common.fxcrt;
...
namespace graphics_objectsCS
{
    class graphics_objects
    {
    ...
    static bool ChangeTextObjectContent()
        {

            string input_file = input_path + "AboutFoxit.pdf";
            using (var doc = new PDFDoc(input_file))
            {
                ErrorCode error_code = doc.Load(null);
                if (error_code != ErrorCode.e_ErrSuccess)
                {
                    Console.WriteLine("The Doc {0} Error: {1}", input_file, error_code);
                    return false;
                }
                try
                {
                    // Get original shading objects from the first PDF page.
                    using (var original_page = doc.GetPage(0))
                    {
                        using (var progressive = 
original_page.StartParse((int)PDFPage.ParseFlags.e_ParsePageNormal, null, false))
                        {
                            PointF pointf = new PointF(92, 762);
                            using (var arr = 
original_page.GetGraphicsObjectsAtPoint(pointf, 10, GraphicsObject.Type.e_TypeText))
                            {
                                for (int i = 0; i < arr.GetSize(); i++)
                                {
                                    using (var graphobj = arr.GetAt(i))
                                    {
                                        using (var textobj = graphobj->GetTextObject())
                                        {
                                            textobj.SetText("Foxit Test");
                                        }
                                    }
                                }
                            }
                            original_page.GenerateContent();
                        }
                    }
                    string output_directory = output_path + "graphics_objects/";
                    string output_file = output_directory + "After_revise.pdf";
                    doc.SaveAs(output_file, (int)PDFDoc.SaveFlags.e_SaveFlagNormal);
                }
                catch (System.Exception e)
                {
                    Console.WriteLine(e.Message);
                }
                return true;
            }
        }
    ...
  }
}
```

3. Can I change the DPI of an embedded TIFF image?

No, you cannot change it. The DPI of the images in PDF files is static, so if the images already exist, Foxit PDF SDK does not have functions to change its DPI.

The solution is that you can use third-party library to change the DPI of an image, and then add it to the PDF file.

***Note**: Foxit PDF SDK provides a function "Image.SetDPIs" which can set the DPI property of an image object. However, it only supports the images that are created by Foxit PDF SDK or created by function "Image.AddFrame", and it does not support the image formats of JPX, GIF and TIF.*

REFERENCES

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

[2] PDF reference 2.0
https://www.iso.org/standard/63534.html

[3] Foxit PDF SDK API reference
sdk_folder/doc/Foxit PDF SDK Dotnet API Reference.html

Note: sdk_folder is the directory of unzipped package.

SUPPORT

Foxit support home link:

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

Sales contact phone number:

Phone: 1-866-680-3668

Email: sales@foxitsoftware.com

Support & General contact:

Phone: 1-866-MYFOXIT or 1-866-693-6948

Email: support@foxitsoftware.com